

webMethods Integration Server Administrator's Guide

Version 10.5

October 2019

This document applies to webMethods Integration Server 10.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2007-2024 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <https://softwareag.com/licenses/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

Document ID: IS-AG-105-20240405

Table of Contents

About this Guide	17
Document Conventions.....	18
Online Information and Support.....	19
Data Protection.....	20
1 The Role of the Administrator	21
What Does an Administrator Do?.....	22
Typical Administrative Responsibilities.....	22
The Integration Server Administrator.....	23
Receiving Administrative Messages from the Server.....	23
The Administrator User.....	23
Adding Backup Administrators.....	23
2 An Overview of the Server	25
The Role of the Server.....	26
Architecture.....	26
How the Server Executes Services.....	30
Integration Server Security.....	31
Integration Server Logging.....	32
Integration Server Caching.....	34
About Integration Server Fixes.....	34
3 Starting and Stopping the Server	37
Starting the webMethods Integration Server.....	38
What Happens When You Start the Server?.....	43
Running Integration Server as a Windows Application vs. a Windows Service.....	44
Increasing File Descriptors on Mac OS X or Other UNIX System.....	46
Changing Settings in the Configuration File custom_wrapper.conf.....	47
Passing Java System Properties to Integration Server.....	51
Shutting Down the Integration Server.....	52
Viewing Active Sessions.....	54
Killing Sessions.....	54
Viewing the Integration Server Process ID.....	54
Restarting the Integration Server.....	55
Server Recovery.....	56
The Java Service Wrapper.....	58
Re-registering a Windows Service after Installing a New Version of the Java Service Wrapper.....	61
4 Running Multiple Integration Server Instances	63
Overview of Integration Server Instances.....	64
Guidelines for Running Multiple Integration Server Instances on the Same Machine.....	64
About Creating a New Integration Server Instance.....	64

About the is_instance Script.....	65
Creating a New Integration Server Instance.....	66
Updating an Integration Server Instance.....	69
Updating Packages on a Server Instance.....	70
Updating Database Properties of a Server Instance.....	71
Deleting Packages from a Server Instance.....	72
Updating Language Packs on a Server Instance.....	73
Deleting a Server Instance.....	74
5 Using the Integration Server Administrator.....	77
What Is the Integration Server Administrator?.....	78
Starting the Integration Server Administrator.....	78
Starting Integration Server Administrator on Windows.....	79
Accessing Integration Server Administrator through My webMethods.....	79
Basic Operations of Integration Server Administrator.....	80
The Server Configuration File.....	81
Software AG Command Central.....	81
6 Managing Users and Groups.....	83
Users and Groups.....	84
Defining a User Account.....	84
Disabling and Enabling User Accounts.....	96
Defining Groups.....	98
7 Configuring the Server.....	105
Viewing and Changing Licensing Information.....	106
Managing the Server Thread Pool.....	109
Managing Server Sessions.....	110
Configuring Outbound HTTP Settings.....	112
Setting Up Aliases for Remote Integration Servers.....	115
Specifying Third-Party Proxy Servers for Outbound Requests.....	118
Configuring Where the Integration Server Writes Logging, Status, and Other Information.....	127
Switching from the Embedded Database to an External RDBMS.....	127
Working with Extended Configuration Settings.....	127
Specifying Character Encoding.....	129
Configuring the JVM.....	129
Specifying the JDK or JRE for Integration Server.....	129
Changing the JVM Heap Size Used by Integration Server.....	130
Publishing and Retracting Information about Integration Server Assets.....	131
Setting a Port for Remote Client JMX Monitoring.....	133
Configuring Integration Server to Accept a Debug Connection During Startup.....	134
Using CORS with Integration Server.....	135
8 Connecting Integration Server to Database Components in an External RDBMS.....	139
Concepts.....	140
Pointing Functions at Connection Pools.....	140
Installing the MySQL Community Edition Database Driver.....	141
Installing the Tibero JDBC Driver.....	141

Creating a Driver Alias Definition.....	142
Creating a Connection Pool.....	142
9 Configuring Ports.....	149
About Ports.....	151
Considerations for Adding Ports.....	154
Adding an HTTP Port.....	155
Adding an HTTPS Port.....	162
About File Polling Ports.....	169
Adding an FTPS Port.....	175
Adding an FTP Port.....	179
Adding an E-Mail Port.....	181
Adding an HTTP Diagnostic Port.....	189
Adding an HTTPS Diagnostic Port.....	194
Suspending an HTTP/HTTPS Port.....	202
Resuming an HTTP/HTTPS Port.....	202
Testing for HTTPS Requests.....	203
Using an FTP/FTPS Port Range.....	203
About the Primary Port.....	204
Deleting a Port.....	205
Editing a Port.....	206
About Enabling/Disabling a Port.....	206
Configuring How Ports Handle Client Certificates.....	207
Adding a Security Provider.....	208
Configuring the Allowed Protocols for JSSE per Port.....	209
Disabling TLS Renegotiation.....	210
Designating an HTTP/S Port as Stateless.....	211
10 Setting Up the Server Log.....	213
Overview of the Server Log.....	214
Specifying Amount and Type of Information to Include in the Server Log.....	215
Specifying Whether to Queue Server Log Entries.....	217
Changing the Default Server Log Location.....	218
Configuring the Server Log to Rotate Based on Size.....	218
Limiting the Number of Server Log Files Kept by Integration Server.....	219
Sending Messages About Critical Issues to E-mail Addresses.....	220
Performing Additional Processing on Log Entries.....	223
Viewing the Server Log.....	223
Changing the Log Displays.....	225
Overriding Logging Level and Server Log Location for a Session.....	227
Globalization and the Server Log.....	228
11 Configuring Integration Server for webMethods Messaging.....	231
11.1 Overview of webMethods Messaging.....	232
Working with Messaging Connection Aliases.....	232
Authenticating Connections to the Universal Messaging Server.....	256
Specifying the Keep-Alive Mode for the Broker Connection.....	258
Synchronizing Broker Clients When the Primary Port for Integration Server Changes.....	261

Configuring Document Stores.....	261
Load Balancing with a Non-Clustered Group of Integration Servers.....	269
12 Configuring Integration Server for JMS Messaging.....	273
12.1 Overview of JMS Messaging Configuration.....	274
Working with JNDI Providers.....	274
Working with JMS Connection Aliases.....	284
Creating Administered Objects.....	307
Monitoring a Connection Factory Object for Changes.....	309
Using SSL with JMS.....	313
Supported JMS Providers.....	314
Adding JMS Provider Client Libraries to Integration Server Classpath.....	319
13 Configuring Integration Server for MQTT Messaging.....	325
Overview of MQTT Support in Integration Server.....	326
Supported MQTT Servers.....	327
Creating an MQTT Connection Alias.....	328
Editing an MQTT Connection Alias.....	331
Enabling and Disabling an MQTT Connection Alias.....	332
Deleting an MQTT Connection Alias.....	333
Specifying a Retry Interval for Failed Connections to the MQTT Server.....	333
14 Using Enhanced Logging for Messaging.....	335
About Enhanced Logging for Messaging.....	336
What Is in Log Entries for Enhanced Logging?.....	336
Using the Messaging Log or the Server Log for Enhanced Logging.....	338
Configuring Enhanced Logging for Messaging.....	339
Viewing the Messaging Log.....	340
15 Setting Up Universal Messaging Client Logging.....	341
Universal Messaging Client Logging.....	342
Configuring Universal Messaging Client Logging.....	342
View the Universal Messaging Client Log.....	343
16 Configuring Endpoint Aliases for Web Services.....	345
About Web Service Endpoint Aliases.....	346
Creating an Endpoint Alias for a Provider Web Service Descriptor for Use with HTTP/S.....	347
Creating an Endpoint Alias for a Consumer Web Service Descriptor for Use with HTTP/S.....	355
Creating an Endpoint Alias for Message Addressing for Use with HTTP/S.....	367
Creating an Endpoint Alias for a Provider Web Service Descriptor for Use with JMS.....	373
Creating an Endpoint Alias for a Consumer Web Service Descriptor for Use with JMS.....	379
Creating an Endpoint Alias for Message Addressing for Use with JMS.....	386
Timestamps in the WS-Security Header.....	392
17 Configuring Reliable Messaging in Integration Server.....	393
Overview of Reliable Messaging.....	394
Using Reliable Messaging in Integration Server.....	395

Configuring Reliable Messaging in Integration Server.....	397
Reliable Messaging Sequence Reports.....	399
Closing a Sequence.....	401
Terminating a Sequence.....	401
Sending an Acknowledgement Request.....	402
18 Configuring Integration Server to Use JWT.....	403
Overview of JWT.....	404
Using JWT with Integration Server.....	405
Support for JWT in Integration Server.....	406
Configuring Integration Server to use JWT.....	407
19 Configuring Integration Server to Use Kerberos.....	413
Overview of Kerberos Usage in Integration Server.....	414
About Kerberos.....	414
Kerberos Delegated Authentication.....	415
Prerequisites to Configuring Kerberos.....	417
Limitations When Using Kerberos Authentication in Integration Server.....	417
Configuring Integration Server to Use Kerberos.....	417
JAAS Contexts for Kerberos.....	420
Troubleshooting Kerberos Configuration.....	421
20 Setting Up HTTP URL Aliases.....	423
Overview.....	424
Creating an HTTP URL Alias.....	425
Enabling Partial Matching of URL Aliases.....	430
Displaying HTTP URL Aliases.....	431
Editing a URL Alias.....	432
Deleting a URL Alias.....	432
Portability of URL Aliases and Possible Conflicts.....	433
21 Using HTTP Interceptors.....	435
Overview of HTTP Interceptors.....	436
Considerations for Creating an HTTP Interceptor.....	437
Creating an HTTP Interceptor.....	438
22 Configuring Integration Server for HTTP Compression.....	439
Overview.....	440
HTTP Request Compression.....	442
HTTP Response Compression.....	443
23 Configuring Integration Server to Connect to an SFTP Server.....	445
Overview of SFTP.....	446
Creating an SFTP Server Alias.....	446
Creating an SFTP User Alias.....	450

24 Configuring Integration Server for Secure Communication.....	457
Overview.....	458
Anatomy of an Integration Server SSL Connection.....	458
Roadmap for Configuring SSL.....	460
Configuring Server-Side SSL Settings.....	463
Controlling Server SSL Security Level by Port.....	464
Storing SSL Information for the Integration Server JVM in a Secure Manner.....	464
Specifying Cipher Suites for Use with SSL.....	466
Usage of CA Certificates: Technical Considerations.....	467
WS-Security and Integration Server.....	468
Using SAML for Web Service Client Authentication.....	468
Requirements for Using SAML for Authentication.....	468
Identifying Trusted STSs to Integration Server.....	469
Accepting SAML2 Tokens at the Transport Level.....	471
25 Setting Up SSL Session Logging.....	473
Overview of the SSL Session Log.....	474
Enabling the SSL Session Logging.....	475
Viewing the SSL Session Log.....	475
Changing the Default SSL Session Log Location.....	475
Managing the SSL Session Log Size.....	476
Avoiding Duplicate Entries in the Log.....	476
26 Using Keystores and Truststores with Integration Server.....	477
Keystores and Truststores.....	478
27 Controlling Access to Resources.....	485
Overview.....	486
Controlling Access to Resources by Port.....	487
Controlling the Use of Directives.....	499
Controlling Access to Resources with ACLs.....	501
Adding Services to a Blacklist.....	512
28 Authenticating Clients.....	515
Overview.....	516
Basic Authentication.....	516
Digest Authentication.....	517
Kerberos Authentication.....	517
Client Certificates.....	518
Using Multiple Client Certificates.....	523
Client Authentication and Access Control.....	525
Accessing Integration Server Data through My webMethods.....	525
29 Customizing Authentication Using JAAS.....	527
Overview.....	528
Using JAAS with Integration Server.....	528

JAAS Configuration File.....	528
Pluggable Authentication Modules (PAMs).....	530
Writing a Custom JAAS Login Module for Integration Server.....	531
JAAS Custom Login Module Example.....	532
30 Master Passwords and Outbound Passwords.....	537
Overview.....	538
Managing Outbound Passwords.....	538
Backing Up Outbound Password and Master Password Files.....	539
Changing the Master Password.....	540
Changing the Expiration Interval for the Master Password.....	540
About the configPassman.cnf File.....	541
Working with Outbound Password Settings.....	542
Working with Master Password Settings.....	543
What to Do if You Lose or Forget Your Master Password.....	546
When Problems Exist with the Master Password or Outbound Passwords at Startup.....	546
E-mail Listeners and Package Replication.....	549
31 Securing Integration Server with CSRF Guard.....	551
What is CSRF?.....	552
How Does Integration Server Prevent CSRF Attacks?.....	552
Understanding CSRF Guard Terminology.....	552
Configuring CSRF Guard in Integration Server.....	554
Limitations when Configuring CSRF Guard in Integration Server.....	556
32 Configuring webMethods Enterprise Gateway.....	557
Overview.....	559
How Enterprise Gateway Works.....	559
Version Interoperability Between Enterprise Gateway Server and Internal Server.....	562
Advantages of Enterprise Gateway over Traditional Third-Party Proxy Servers.....	563
About Denial of Service Protection.....	564
About Mobile Application Protection.....	565
About Mobile Data Synchronization.....	565
About SQL Injection Protection.....	565
About Antivirus Scan Filter.....	566
About Custom Filter.....	568
Clustering in the Enterprise Gateway Configuration.....	568
Setting Up an Enterprise Gateway.....	569
Configuring the Enterprise Gateway Ports.....	571
Connecting Your Internal Server to an Enterprise Gateway Server.....	576
Viewing Connections to the Enterprise Gateway Registration Port.....	580
Performing Client Authentication on Enterprise Gateway Server.....	581
Working with Enterprise Gateway Rules.....	582
Specifying Alert Options.....	589
Preventing Denial of Service Attacks.....	591
Controlling Use of Mobile Applications.....	594
Frequently Asked Questions about Enterprise Gateway.....	596

33 Configuring OAuth	601
What Is OAuth?.....	602
Using OAuth with Integration Server.....	602
OAuth Client Types.....	604
Authorization Grant Types Supported by Integration Server.....	604
The Integration Server OAuth Services.....	610
Important Considerations for Using OAuth Features.....	612
Configuring Integration Server for OAuth.....	612
About Using Integration Server as the Resource Server.....	630
Using an External Authorization Server.....	630
34 Configuring a Central User Directory or LDAP	635
Before You Begin.....	636
Overview of How Integration Server Works with Externally Defined Users and Groups..	636
Configuring Central User Management.....	638
Overview of Using LDAP.....	640
Configuring the Server to Use LDAP.....	641
Considerations for User Accounts and Groups.....	647
About Granting Administrator Privileges to External Users.....	649
Granting Developer Privileges to External Users.....	650
Granting Access to Services and Files to External Users.....	651
35 Managing Packages	653
Using Packages.....	654
How the Server Stores Package Information.....	657
Finding Information about Your Packages.....	660
Working with Packages.....	667
Copying Packages from One Server to Another.....	673
Using a Package Class Loader.....	697
Hot Deployment of Packages.....	697
Automatic Package Deployment.....	701
36 Managing Services	703
About Services.....	704
Fully Qualified Service Names.....	704
Finding Information about Services and Folders.....	705
Manually Adding a Service to the Server.....	707
Testing Services.....	707
Canceling and Killing Threads Associated with a Service.....	708
Running Services When Packages Are Loaded, Unloaded, or Replicated.....	710
Running Services in Response to Specific Events.....	712
Managing Global Variables.....	712
37 Scheduling Services	715
Overview.....	716
Scheduling a User Task.....	716

Viewing Scheduled User Tasks.....	721
Updating Scheduled User Tasks.....	722
Suspending User Tasks.....	723
Resuming Suspended User Tasks.....	724
Canceling a Scheduled User Task.....	725
Viewing the Scheduled System Tasks.....	725
Simple Repeating Option.....	726
Complex Repeating Option.....	727
Target Node Options.....	730
How Transitioning to or from Daylight Savings Time Affects Scheduled Tasks.....	732
38 Caching Service Results.....	733
What Is Caching?.....	734
When Are Cached Results Returned?.....	734
Using a Public Cache for Service Results Caching.....	735
Resetting the Cache.....	736
Monitoring Service Cache Usage.....	737
Viewing Service Results in a Public Cache.....	737
39 Configuring Guaranteed Delivery.....	739
About Guaranteed Delivery.....	740
Configuring the Server for Guaranteed Delivery.....	741
Administering Guaranteed Delivery.....	745
40 Configuring Ehcache on Integration Server.....	749
What is Ehcache?.....	750
Caching Configurations.....	750
Understanding Caches and Cache Managers.....	753
Cache Manager Configuration Files.....	755
Installing, Viewing, and Changing the Terracotta License.....	756
Configuring an On-Heap Cache.....	758
Configuring a BigMemory Cache.....	760
Configuring a Distributed Cache.....	763
Making a Cache Searchable.....	772
Working with Cache Managers.....	775
Working with Caches.....	783
Logging Ehcache Activity.....	797
41 Configuring the Enhanced XML Parser.....	799
What Is the Enhanced XML Parser?.....	800
How Does the Enhanced XML Parser Work?.....	800
When Is the Enhanced XML Parser Used?.....	802
Configuring the Enhanced XML Parser.....	803
Setting the Partition Size.....	806
Viewing Peak Usage Statistics.....	807
42 Configuring WebSockets.....	809

Overview.....	810
How WebSocket Works.....	810
Setting Up Integration Server for WebSockets.....	811
Configuring a WebSocket Ports.....	811
Configuring a WebSocketSecure Ports.....	814
Viewing WebSocket Server Endpoints for a WebSocket Port.....	817
WebSocket Server Sessions Screen.....	818
Limitations When Using WebSocket Protocol in Integration Server.....	818
43 Locking Administration and Best Practices.....	821
Introduction.....	822
Choosing Local Server Locking or VCS Integration Locking.....	822
Disabling and Re-enabling Locking.....	822
Best Practices.....	824
44 Managing webMethods Messaging Triggers.....	827
Introduction.....	828
Managing Document Retrieval.....	828
Managing Document Processing.....	838
Limiting Server Threads for webMethods Messaging Triggers.....	846
Cluster Synchronization for webMethods Messaging Trigger Management.....	848
Modifying webMethods Messaging Trigger Properties.....	851
Managing Trigger Service Retries and Shutdown Requests.....	852
Delaying Polling Requests for webMethods Messaging Triggers.....	853
Serial Triggers Migrated to Integration Server 10.3 or Later from Earlier Versions.....	857
45 Managing JMS Triggers.....	859
45.1 Introduction to JMS Trigger Management.....	860
Searching for JMS Triggers.....	860
About JMS Trigger Status and State.....	861
Controlling Thread Usage for JMS Triggers.....	865
Configuring Integration Server Session Reuse.....	869
Configuring JMS Session Reuse.....	869
Delaying Polling Requests for Concurrent JMS Triggers.....	869
What Happens When JMS Triggers Fail to Start?.....	872
About WS Endpoint Triggers.....	875
46 Managing MQTT Triggers.....	879
Introduction to MQTT Trigger Management.....	880
MQTT Trigger State and Status.....	880
Configuring Integration Server Session Reuse for MQTT Triggers.....	882
Automatic Retry for Starting MQTT Triggers.....	883
47 Using a Document History Database with Exactly-Once Processing.....	885
Overview.....	886
Removing Expired Entries from the Document History Database.....	886
Viewing Exactly-Once Processing Statistics.....	887

Clearing Exactly-Once Processing Statistics.....	887
48 Using Integration Server to Manage XA Transactions.....	889
Overview of XA Transaction Management.....	890
Configuring XA Options in Integration Server.....	894
Manually Resolving a Transaction.....	897
49 Content Handlers in Integration Server.....	901
How Integration Server Uses Content Handlers.....	902
How Integration Server Chooses a Content Handler for an HTTP Request.....	902
How Integration Server Chooses a Content Handler for an FTP Request.....	902
Content Handlers for HTTP and FTP Requests.....	902
About Content Handlers for HTTP Responses.....	908
About Content Handlers for FTP Responses.....	909
50 Class Loading in Integration Server.....	911
How Class Loading Works in Integration Server.....	912
Adding Classes to the Server Classpath.....	917
Where to Put Your Classes and Jar Files.....	922
Accelerating Class Loading.....	923
51 Whitelist Filtering in Integration Server.....	925
About Whitelist Filtering in Integration Server.....	926
Enabling or Disabling Whitelist Class Filtering.....	926
Integration Server Whitelist Classes File.....	926
Creating a Package Whitelist Classes File.....	927
Logging Classes that Are Not on the Whitelist.....	928
Integration Server Blacklist Classes File.....	928
52 Quiescing the Server for Maintenance.....	929
Overview.....	930
Specifying the Quiesce Settings.....	933
Quiescing Integration Server.....	933
Exiting Quiesce Mode.....	935
Deploying Quiesce Mode Configuration.....	936
53 Diagnosing the Integration Server.....	937
Introduction.....	938
Configuring the Diagnostic Port.....	938
Using the Diagnostic Utility.....	939
Starting the Integration Server in Safe Mode.....	942
When the Server Automatically Places You in Safe Mode.....	943
Generating Thread Dumps.....	943
54 Debugging Service Exceptions Using the Error Log.....	947
Introduction.....	948

Controlling the Level of Exception Logging Detail.....	948
Displaying the Error Log.....	948
Interpreting the Error Log.....	948
55 Using Integration Server with Docker.....	951
Overview of Docker and Integration Server.....	952
About the is_container Script.....	953
Prerequisites for Building a Docker Image.....	954
Specifying Services to Expose to Consumers in webMethods Cloud.....	955
Building the Docker Image for an Integration Server Instance.....	955
Loading a Docker Image to an On-Premise Docker Registry.....	961
Pushing a Docker Image to an On-Premise Docker Registry.....	962
Running the Docker Image in an On-Premise Docker Container.....	963
Pushing the Docker Image to Integration Cloud.....	969
Running the Docker Image in Integration Cloud.....	970
Stopping a Docker Container for Integration Server.....	970
Using a Configuration Variables Template with a Docker Image.....	970
56 Integration Server Administrator API.....	971
Introduction to the Administrator API.....	972
Authentication and Authorization.....	972
REST URL Structure.....	972
CRUD Operations.....	973
Administrative Actions.....	974
Credentials.....	974
Expanding Responses.....	974
Media Type for Responses.....	976
Media Type for Request Payloads.....	976
Getting Started.....	976
Controlling Access to the Administrator API.....	977
Logging Client Errors.....	977
Including a Stack Trace in the Response Body.....	978
Integration Server Administrator API Operations.....	978
57 Simulating Metering in Integration Server.....	981
About webMethods Metering.....	982
About the Metering Simulator.....	983
Enabling the Metering Simulator.....	984
58 Using Command Central to Manage Integration Server.....	985
An Overview of Command Central.....	986
Integration Server Instance Management.....	986
Accessing Integration Server Administrator through Command Central.....	992
Monitoring Integration Server Instances.....	994
Integration Server Configuration Types.....	996
Lifecycle Actions for Integration Server.....	1003
Commands that Integration Server Supports.....	1003
Using Unix Shell Scripts to Change Connection Credentials for Managed Products.....	1004

Command Central Command Line Tool Upgrade.....	1005
A Integration Server Deployment Checklist.....	1007
Introduction.....	1008
Stage 1: Installation.....	1008
Stage 2: Basic Configuration.....	1009
Stage 3: Setting Up Users, Groups, and ACLs.....	1010
Stage 4: Publishing Packages.....	1011
Stage 5: Installing Run-Time Classes.....	1012
Stage 6: Preparing Clients for Communication with the Server.....	1012
Stage 7: Setting Up Security.....	1013
Stage 8: Startup and Test.....	1014
Stage 9: Archive Sources.....	1015
B Server Configuration Parameters.....	1017
Introduction.....	1019
watt.adapters.....	1019
watt.adminapi.....	1019
watt.art.....	1020
watt.broker.....	1021
watt.brokerCoder.....	1021
watt.cachedirective.....	1021
watt.cds.....	1022
watt.client.....	1022
watt.config.....	1023
watt.core.....	1023
watt.debug.....	1029
watt.debug2.....	1031
watt.frag.....	1032
watt.infradc.....	1032
watt.net.....	1032
watt.security.....	1048
watt.server.....	1053
watt.ssl.....	1177
watt.ssh.....	1178
watt.tx.....	1179
watt.um.....	1180
watt.wm.tnextdc.....	1181
watt.wmcloud.....	1181
C Environment Variables for Use with Docker.....	1183
Environment Variables.....	1184
D FIPS 140-2 Compliance.....	1189
FIPS 140-2 Compliance.....	1190
E Using NTLM Authentication when Integration Server Acts as the Client.....	1191

Overview.....	1192
Using Java-Based NTLM Support.....	1192
Using Native NTLM Support via Integrated Windows Authentication.....	1193
F Removing User Data from Integration Server.....	1197
Removing User Data.....	1198
Removing References to a User Account.....	1198
Removing References to Email Addresses.....	1201
Removing Personal Data from Log Files.....	1202
G Wireless Communication with the Integration Server.....	1209
Overview.....	1210
How Does the Integration Server Communicate with Wireless Devices?.....	1210
Using URLs for Wireless Access to the Integration Server.....	1211
H Masking Session IDs in Integration Server Session and Server Logs.....	1215
I Server Log Facilities.....	1217
About Server Log Facilities.....	1218
Integration Server.....	1218
WmJSONAPI Package.....	1224
WmXSLT Package.....	1224
Flat File.....	1224

About this Guide

- Document Conventions 18
- Online Information and Support 19
- Data Protection 20

This guide is for the administrator of a webMethods Integration Server. It provides an overview of how the server operates and explains common administrative tasks such as starting and stopping the server, configuring the server, setting up user accounts and security, and managing packages and services.

Note:

This guide describes features and functionality that may or may not be available with your licensed version of webMethods Integration Server. For information about the licensed components for your installation, see the **Settings > License** page in the Integration Server Administrator.

Integration Server provides a subset of the functionality available in webMethods Microservices Runtime. If your webMethods Integration Server is equipped with a Microservices Runtime license, you can use features and functionality available in Microservices Runtime. For more information about Microservices Runtime, see *Developing Microservices with webMethods Microservices Runtime*.

Unlike Integration Server, Microservices Runtime does not allow multiple instances, does not make use of OSGi, and does not use the Java Service Wrapper. Consequently, the directory structures of Integration Server and Microservices Runtime are different and some configuration differences exist due to the fact that Microservices Runtime does not use the Java Service Wrapper for passing properties to Microservices Runtime.

When reviewing the *webMethods Integration Server Administrator's Guide*, keep in mind that references to *Integration Server_directory /instances/instanceName* for Integration Server correspond to the *Integration Server_directory* in Microservices Runtime. Differences between Integration Server and Microservices Runtime are noted in the *webMethods Integration Server Administrator's Guide* when they exist.

Note:

The directory and configuration differences apply to a Microservices Runtime only. That is, an Integration Server equipped with an Microservices Runtime license and an Integration Server without the Microservices Runtime license have the same directory structure, use OSGi, and makes use of the Java Service Wrapper.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.

Convention	Description
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 The Role of the Administrator

■ What Does an Administrator Do?	22
■ Typical Administrative Responsibilities	22
■ The Integration Server Administrator	23
■ Receiving Administrative Messages from the Server	23
■ The Administrator User	23
■ Adding Backup Administrators	23

What Does an Administrator Do?

In an IS environment, the administrator is responsible for installing, configuring, and maintaining the webMethods Integration Server. He or she is also responsible for ensuring the server is secure, available to clients, and running at peak performance. Usually, one person is appointed as the administrator, although most sites identify at least one other person to act as a backup.

Typical Administrative Responsibilities

If you are the webMethods Integration Server administrator for your site, you might be involved in some or all of the following activities.

- **Installing and upgrading the server**, which includes tasks such as equipping the server computer with appropriate hardware and software, downloading and installing the server program, and implementing upgrades as needed.
- **Starting and stopping the server**, which includes shutting down the server when necessary (e.g., for routine maintenance or reconfiguration) and restarting it afterwards. It also includes performing your site's standard recovery procedures following a hardware or software failure of the server computer. For information about these activities, see [“Starting and Stopping the Server” on page 37](#).
- **Configuring server settings**, which includes setting basic operating parameters such as the maximum session limits, log file options, and port assignments. For information about these activities, see [“Configuring the Server” on page 105](#).
- **Administering users and groups**, which includes defining user names and passwords for authorized users and assigning them to groups. For information about this task, see [“Managing Users and Groups” on page 83](#). Alternatively, you can configure the server to acquire user and group information from an external system (e.g., LDAP). For more information, see [“Configuring a Central User Directory or LDAP” on page 635](#).
- **Administering server security**, which includes identifying other administrators, assigning access controls to individual services, and configuring the server's use of digital certificates. For more information about this task, see [“Customizing Authentication Using JAAS” on page 527](#), [“Configuring Integration Server for Secure Communication” on page 457](#), [“Customizing Authentication Using JAAS” on page 527](#), and [“Configuring Integration Server for Secure Communication” on page 457](#).
- **Managing packages and services**, which includes tasks such as activating/deactivating/copying packages and updating services and/or packages as necessary. For more information about this task, see [“Managing Packages” on page 653](#) and [“Managing Services” on page 703](#).
- **Administering multiple instances of the server**, which includes performing all or some of the activities listed above to manage two or more Integration Servers running on the same machine. For more information about running multiple instances of Integration Server on the same machine, see [“Running Multiple Integration Server Instances” on page 63](#).

The Integration Server Administrator

The Integration Server Administrator is the utility you use to accomplish administrative tasks. You use it to monitor server activity, examine log information, add users, enable/disable services, and adjust the server's performance features. For information about the Integration Server Administrator, see [“Using the Integration Server Administrator ” on page 77](#).

Receiving Administrative Messages from the Server

The Integration Server issues e-mail messages for a variety of failure conditions (for example, internal errors, binding errors, and transaction manager errors). As an administrator, you are the one who should receive these messages and take appropriate action when errors occur.

To ensure that you (or an appropriate alternate) receive messages from the server, you must set the **E-mail Notification** parameters using the Integration Server Administrator as described in [“Specifying an E-Mail Address and SMTP Server for Error Messages” on page 744](#).

The Administrator User

Every Integration Server is installed with a predefined user account called "Administrator." By default, this user is the only one who can perform administrative tasks with the Integration Server Administrator.

The Administrator's Password

The predefined password assigned to the Administrator user account is "manage".

Important:

Change this password *immediately* after installing the webMethods Integration Server. Otherwise, your server will be vulnerable to anyone who knows the default password that webMethods installs on its servers. Change the predefined passwords for the Developer and Replicator user accounts as well. Software AG also recommends a best practice of creating an individual account for each administrator and developer instead of using the predefined accounts. Disable the predefined accounts after creating individual accounts

When assigning a password, make it something that is difficult to guess. For example, make it a mixture of upper- and lowercase letters, numbers, and special characters. Passwords cannot be null values or empty strings. Do not use a name, a phone number, your license plate, your social security number, or other generally available information. Do not write passwords down. Do not tell anyone the password unless you are sure of that person's identity.

For instructions on changing a password, see [“Adding an Administrator User” on page 88](#).

Adding Backup Administrators

It is a good idea to designate at least one individual as a "backup administrator," who can administer the Integration Server when you are not available.

To add a backup administrator to your server, create a regular user account for the user (if he or she does not already have one); then add that user account to the "Administrators" group.

Only members of the "Administrators" group can use the Integration Server Administrator. For information about creating user accounts and adding them to groups, see ["Managing Users and Groups" on page 83](#).

Note:

If you use an external directory for user and group information, see ["About Granting Administrator Privileges to External Users" on page 649](#) for information about adding administrators.

2 An Overview of the Server

■ The Role of the Server	26
■ Architecture	26
■ How the Server Executes Services	30
■ Integration Server Security	31
■ Integration Server Logging	32
■ Integration Server Caching	34
■ About Integration Server Fixes	34

The Role of the Server

The webMethods Integration Server hosts packages that contain services and related files. The Integration Server comes with core Integration Server packages. For example, all Integration Server instances include packages that contain built-in services that your developers might want to invoke from their services or client applications and services that demonstrate some of the features of the Integration Server. You can create additional packages to hold the services that your developers create. Your developers can create services that perform functions, such as integrating your business systems with those of your partners, retrieving data from legacy systems, and accessing and updating databases.

webMethods Integration Server provides an environment for the orderly, efficient, and secure execution of services. It decodes client requests, identifies the requested services, invokes the services, passes data to them in the expected format, encodes the output produced by the services, and returns output to the clients.

Additionally, the server authenticates clients, verifies that they are authorized to execute the requested service, maintains audit-trail logs, and promotes throughput using facilities such as service result caching.

About Integration Server Instances

webMethods Integration Server allows you to create and run multiple instances of the server on one machine. When you install webMethods Integration Server, you specify a name for the initial instance. The default name of the initial instance is “default”. The Software AG installer creates the instance under the *Software AG_directory* \IntegrationServer\instances directory. You can create additional instances of Integration Server using the scripts provided by Integration Server or through Software AG Command Central.

Each instance has a home directory under *Software AG_directory* \IntegrationServer\instances that contains its own packages, configuration files, log files, and updates. You administer and apply packages and updates to each Integration Server instance separately. You can apply the latest fixes using the Software AG Update Manager.

The *Software AG_directory* \IntegrationServer directory is the parent directory for all server instances you create. It contains common and shared files that all server instances use, such as common jar files and fixes. For information about creating and running multiple Integration Server instances, see [“Running Multiple Integration Server Instances” on page 63](#).

Note:

Unless otherwise specified, the terms Integration Server and “the server” in this guide refer to an Integration Server instance.

Architecture

The Integration Server listens for client requests on one or more ports. You can associate the type of protocol that the server uses for each port. The server supports HTTP, HTTPS, FTP, FTPS, and e-mail ports.

Most clients use an HTTP or HTTPS port for communication with the server. Because the server supports both HTTP and HTTPS, it can listen on an HTTP port for non-secure client requests and an HTTPS port for secure requests.

Note:

Unlike HTTP, FTP, and e-mail, HTTPS and FTPS provide for secure data transmission. They do this through encryption and certificates. Without HTTPS or FTPS, unauthorized users might be able to capture or modify data, use IP spoofing to attack servers, access unauthorized services, or capture passwords. If you must pass passwords, make sure the back-end application has minimal privileges.

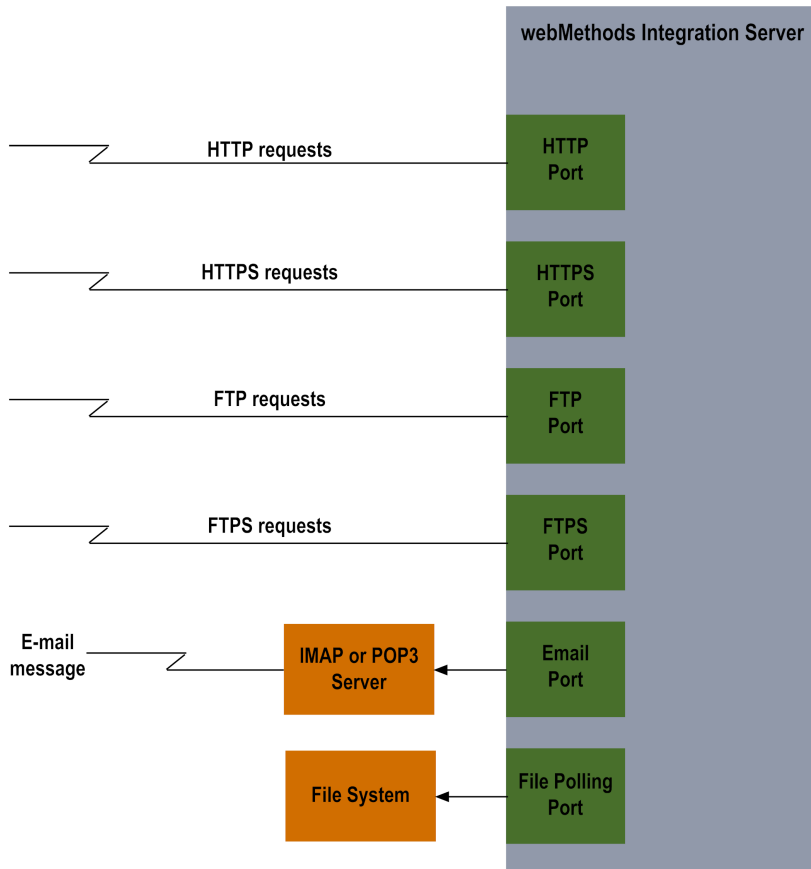
A typical use for an FTP or FTPS port is to get a directory listing, change to the directory that contains the service you want to invoke, put a file that contains input to the service, and run the service. The server returns the output from the service to the directory in which the service resides. Use an e-mail port to receive requests through an e-mail server, such as POP3 or IMAP.

You can define as many ports as you want on each Integration Server instance. The default server instance has an HTTP port at 5555.

Note:

The default server instance also defines a diagnostic port at 9999. The diagnostic port uses the HTTP protocol and provides you access to the Integration Server when it is unresponsive. For more information about the diagnostic port, see [“Diagnosing the Integration Server” on page 937](#).

The Server Listens for Requests on Ports that You Specify



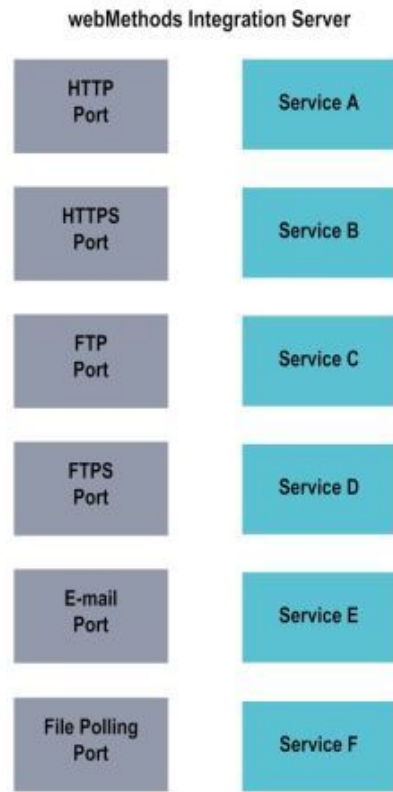
There may be times when you want to use the standard port numbers used by web servers: port 80 for HTTP requests and port 443 for HTTPS requests. If your Integration Server runs on a Windows system, this is not a problem. However, if your Integration Server runs on a UNIX system, using a port number below 1024 requires that the server run as "root." For security reasons, Software AG discourages this practice. Instead, run your Integration Server using an unprivileged user ID on a high number port (for example 1024 or above) and use the port remapping capabilities present in most firewalls to move requests to the higher numbered ports.

Services

Client requests involve executing one or more services. The server maintains successfully loaded services as runnable objects within the server's program space.

When you initialize the server, the server loads the services that are contained in enabled packages into memory. When you or another administrator enable a disabled package, the server loads services that are in that package.

Services Execute within the Integration Server's Virtual Machine



When a client invokes a service, that service runs as a thread within the Integration Server program. Depending on what function the service is to accomplish, it can also create additional threads to perform tasks simultaneously.

Retrieving Data for Services

Tasks that services perform often include retrieving data from data sources. The server can retrieve data (for example, XML and HTML data) from local data sources or by issuing HTTP, HTTPS, FTP, FTPS, e-mail, or file polling requests to resources such as web servers and JDBC-enabled databases.

There are a number of methods you can use to send files from a client to the Integration Server. The Integration Server provides the following automated mechanisms:

- Post a file to a service via HTTP or HTTPS.
- FTP a file to a service.
- Submit a file to a service via a file polling port.
- E-mail a file to a service as an attachment.

Note:

If you use Trading Networks, you can send some files, specifically flat files, directly to Trading Networks. For more information about how Trading Networks processes flat files, see the

"Defining and Managing Flat File Document Types" chapter in *webMethods Trading Networks Administrator's Guide*.

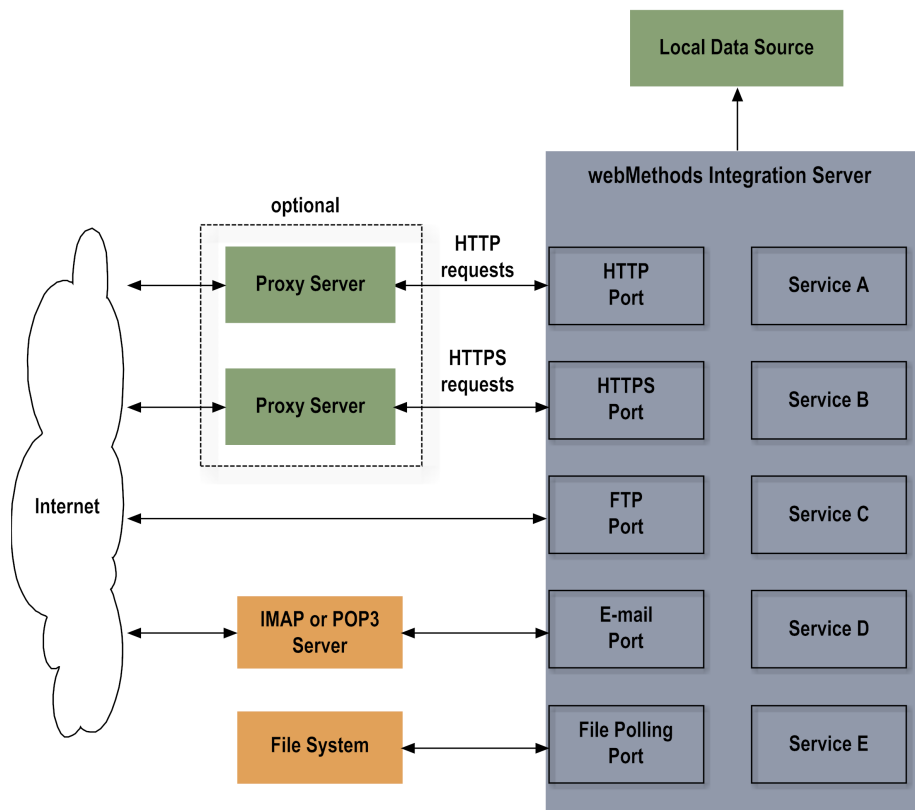
When a client submits a file to the Integration Server, the server uses the appropriate content handler to parse the contents of the file and pass them to the target service.

For all transmission methods except file polling, the client specifies the service to be executed. For file polling, the server always executes the service associated with the file polling port.

For more information about sending and receiving XML files, see *webMethods Service Development Help*. For more information about sending and receiving flat files, see the *Flat File Schema Developer's Guide*. Refer to the *webMethods Integration Server Built-In Services Reference* for information about services you can invoke from the service you write.

When the server uses HTTP or HTTPS to access data from external data sources, you can optionally route the requests through a proxy server.

The Server Gets Data from Local Resources or Resources on the Internet



How the Server Executes Services

When the Integration Server receives a request from a client, it performs the following actions:

1. The server authenticates the client.

2. If a session already exists for the client, the server uses the existing session. If one does not exist, the server creates a session.
3. The server determines the content-type of the service request so it can prepare data for the requested service.
4. The server uses the supplied service name to look up the service.
5. The server determines whether access to the requested service is being controlled based on the port on which the request came in. If there is no restriction, the server continues with the execution of the service.
6. The server checks whether the requested HTTP method is allowed for the service. If it is not, the server sends an back an error message, else the server continues with the execution of the service.
7. The server looks up the Access Control List (ACL) for the service and determines whether the client is to be granted access to the service. If the ACL indicates that the client is allowed to access the service, the server continues with the execution of the service.
8. If auditing is enabled, the server adds an entry to the Audit Log to mark the start of the request.
9. The server starts gathering service statistics for the service.
10. The server checks to see if the results for this service are in the service-results cache. If service results are cached and the inputs for the cached results match the inputs for this request, the server returns the cached results. If matching results are not cached, the server invokes the service. If the service is a flow service, which can consist of several services, it invokes each service in the flow.

Note:

For each service in a flow, the server performs steps 6 through 11.

11. The server ends the gathering of server statistics for the service.
12. If auditing is enabled, the server adds an entry to the Audit Log to mark the end of the request.
13. The server encodes the service results as specified by the content type.
14. The server returns the results to the client.

Integration Server Security

Integration Server's security mechanisms prevent its unauthorized administration, prevent data from being intercepted during transmission, and protect Integration Server services from unauthorized access. You can configure Integration Server to:

- Use an Enterprise Gateway Server to intercept requests from external clients before passing the requests to your Integration Server. This allows you to isolate Integration Server behind an internal firewall.
- Require clients to present valid credentials (i.e., user name and password or a client certificate) to authenticate a connection.

- Authorize access to individual services by user groups, through use of Access Control Lists (ACLs) that you associate with a service. For the greatest security, associate *all* services with an ACL.
- Provide transport-level security through Secure Sockets Layer (SSL), and message-level security for web services through WS-Security.
- Digitally sign documents and verify digital signatures.
- Control access to services based on the port through which a service request is received.
- Restrict who can access Integration Server Administrator, and who can use Software AG Designer to connect to the Integration Server.
- Require clients to present valid user names (with passwords) that have Administrator privileges before allowing access to the Integration Server Administrator functions.
- Simplify security administration by storing Integration Server SSL certificates and private keys in industry-standard keystore files.
- Allow different client certificates to be used for different connections.

Integration Server security also depends on the security of its underlying operating system. Make sure you do the following:

- Follow all vendor recommendations for secure configuration.
- Remove unnecessary network services that may contain security flaws, such as telnet.
- Regularly check for and install updates and patches from the operating system vendor that might affect security.

See your operating system's documentation for instructions on accomplishing these tasks.

For information about security auditing, refer to the *webMethods Audit Logging Guide*.

Integration Server Logging

Logging for the platform provides important data you need to monitor platform activity and correct problems. Integration Server maintains this logging data. For complete information and instructions about working with logging data, see the *webMethods Audit Logging Guide*, "[Setting Up the Server Log](#)" on page 213, and "[Setting Up Universal Messaging Client Logging](#)" on page 341.

Logging, Data Protection, and Privacy

Integration Server includes personally identifiable information (PII) such as user names, email addresses, and client IP addresses in the logs. Integration Server includes PII in logs for purposes of auditing, monitoring activity with the server, and diagnosing and correcting problems. The length of time that Integration Server stores log data depends on the log:

Log	Data duration
Server log	<p>Data remains in the server log for the duration of the existence of the log file which depends on the <code>watt.server.serverlogFilesToKeep</code> server configuration parameter. This parameter limits the number of server log files that Integration Server maintains on the file system. When Integration Server reaches the limit for the number of server log files, Integration Server deletes the oldest archived server log file each time Integration Server rotates the server log.</p> <p>For more information about <code>watt.server.serverlogFilesToKeep</code>, see “Limiting the Number of Server Log Files Kept by Integration Server” on page 219.</p> <p>For information about removing data from the Integration Server server log, see “Removing Personal Data from Log Files” on page 1202.</p>
File-based audit log	<p>Data remains in the log for the duration of the log file existence which depends on the value of <code>watt.server.audit.logFilesToKeep</code> server configuration parameter. This parameter limits the number of audit log files that Integration Server maintains on the file system for each logger. When the number of log files for an audit logger reaches the established limit, Integration Server deletes the oldest audit log file.</p> <p>For more information about <code>watt.server.audit.logFilesToKeep</code>, see “Server Configuration Parameters” on page 1017.</p> <p>For information about removing data from an Integration Server audit log, see “Removing Personal Data from Log Files” on page 1202.</p>
Audit log that writes to a database	<p>Data remains in the database table for the log in the <code>ISCoreAudit</code> database until it is removed.</p> <p>For information about removing data from an Integration Server audit log, see “Removing Personal Data from Log Files” on page 1202.</p>
Configuration variables log	<p>The length of time that data remains in the configuration variables log depends on whether the log is for an on-premises Microservices Runtime or a Microservices Runtime running in a Docker container. The configuration variables log is overwritten at startup of an on-premises Microservices Runtime. The configuration variable log for a Microservices Runtime running in a Docker container is destroyed when the container is destroyed.</p> <p>For information about removing data from a configuration variables log, see “Removing Personally Identifiable Information from the Configuration Variables Log” on page 1206.</p>
Wrapper log	<p>Data remains in the wrapper log for the duration of the log file existence which depends on the value of the <code>wrapper.logfile.maxfiles</code> property for the Java Service Wrapper. For the <code>wrapper.conf</code> file delivered with</p>

Log**Data duration**

Integration Server, this property is set to 5. For more information about this property, see [“Logging Properties” on page 60](#).

For information about removing data from the wrapper log, see [“Removing Personally Identifiable Information Logged by Axis2, Kerberos, SAML, and other Third Party Libraries” on page 1207](#)

SSL session log

Data remains in the inboundSSLSessions.log until it is removed. The `watt.net.ssl.server.sessionlog.maxFileSize` server configuration parameter limits the size of the inboundSSLSessions.log file. When the file reaches the maximum size limit, Integration Server renames the file to `sslsession_<DATE(YYYYMMDD)>_TIME(HHMMSS).log` and creates a new inboundSSLSessions.log file.

For more information about `watt.net.ssl.server.sessionlog.maxFileSize`, see [“Server Configuration Parameters” on page 1017](#).

For information about removing data from the inbound SSL session log, see [“Removing Personally Identifiable Information from the SSL Session Log” on page 1206](#).

Integration Server Caching

Caching is an optimization feature that can improve the performance of services by maintaining frequently used data in memory. Caching can significantly improve response time of services that retrieve information from high-traffic web servers and databases.

Integration Server uses the caching capabilities of Ehcache to provide:

- **Service results caching.** When you enable services to cache results, Integration Server saves the service invocation results in the cache for a specified period of time. While the results are in cache, rather than re-invoking the service, Integration Server can quickly retrieve the service results for subsequent clients' requests for the service. For more information about service caching, see [“Caching Service Results” on page 733](#).
- **Data caching.** With Ehcache, you can define individual caches in which services can cache data. Services that want to cache data do so using the caching services provided in the `pub.cache` folder. Ehcache enables you to cache virtually any kind of object. Additionally, it provides advanced features such as off-heap caching and distributed caching. For more information about using Ehcache, see [“Configuring Ehcache on Integration Server ” on page 749](#).

About Integration Server Fixes

Software AG delivers fixes for Integration Server via the Software AG Update Manager. In addition to including resolutions to product defects, a fix may also include enhancements to existing functionality and features. Any of these items can result in new or changed built-in services, APIs, or configuration parameters. Review the fix contents carefully before applying the fix.

The fix readme file serves as the documentation for the functionality delivered in a fix. The released product documentation is not typically updated for fix functionality. In the event that the released product documentation is updated for functionality delivered in a fix, the refreshed product documentation will include a note identifying the fix in which the functionality was made available.

3 Starting and Stopping the Server

- Starting the webMethods Integration Server 38
- What Happens When You Start the Server? 43
- Running Integration Server as a Windows Application vs. a Windows Service 44
- Increasing File Descriptors on Mac OS X or Other UNIX System 46
- Changing Settings in the Configuration File custom_wrapper.conf 47
- Passing Java System Properties to Integration Server 51
- Shutting Down the Integration Server 52
- Viewing Active Sessions 54
- Killing Sessions 54
- Viewing the Integration Server Process ID 54
- Restarting the Integration Server 55
- Server Recovery 56
- The Java Service Wrapper 58
- Re-registering a Windows Service after Installing a New Version of the Java Service Wrapper 61

Starting the webMethods Integration Server

The webMethods Integration Server must be running in order for clients to execute services. If you are using the server in a development environment, it must be running in order for your developers to build, update, and test services using the Software AG Designer.

Keep the following points in mind when starting Integration Server:

- Integration Server stores some configuration files and log files on disk. Make sure that there is enough free disk space on the Integration Server machine to accommodate these files. Running out of disk space can affect performance and can lead to errors.
- Make sure that the external database to which Integration Server connects is available before starting Integration Server. If the database is unavailable at Integration Server startup, some components, such as the scheduler, will not be initialized and will not work properly.

For information about starting Microservices Runtime, see *Developing Microservices with webMethods Microservices Runtime*.

Starting an Integration Server Instance on Windows

Start an instance of Integration Server from the Windows Start menu as described below.

Note:

If you are running the Windows Server 2008 r2 operating system or Windows Server 2012 with the User Account Control security feature enabled, you must start Integration Server with full Administrator privileges. To start Integration Server with full Administrator privileges, navigate to the list of programs or applications for **Software AG**. For example, in Windows Server 2008, navigate to **All Programs > Software AG**. Right-click **Start *instanceName***, and select the **Run as Administrator** option. If you are not logged into the operating system with Administrator privileges, you will be prompted to supply Administrator credentials.

➤ To start Integration Server on Windows

1. Click **Start**.
2. In the **All Programs** menu, click the **Software AG** folder.
3. Click the **Start Servers** folder.
4. Click **Start Integration Server**.
5. Click **Start Integration Server *instanceName***.

Note:

If your Integration Server has been configured to request a master password for outbound password encryption, you will be prompted for this password in a popup window or from

the server console. Refer to [“Managing Outbound Passwords” on page 538](#) for more information about this password.

Starting Integration Server on UNIX

Keep the following points in mind when starting Integration Server on UNIX using the `startup.sh` script file:

- Run the `startup.sh` script only when logged in as a non-root user. Running the script as root might reduce the security of your system.
- The server can consume more files and sockets on a UNIX system than on other systems. Therefore, if you are running the server on a UNIX system, Software AG recommends that run it with at least 1024 file descriptors. You can increase the number of available file descriptors by entering the following command from the UNIX command prompt before starting the server:

```
ulimit -n number
```

➤ To start Integration Server on UNIX

1. Navigate to the following directory:

```
Software AG_directory \profiles\IS_instance_name\bin
```

where *instance_name* is the name of the Integration Server instance.

Note:

The `startup.bat/sh` and `shutdown.bat/sh` scripts contained in the *Integration Server_directory* \instances\instance_name\bin directory are deprecated. Software AG recommends the use of the scripts contained in the *Software AG_directory* \profiles\IS_instance_name\bin directory. If you will manage Integration Server through Command Central, you *must* use the scripts located in the *Software AG_directory* \profiles\IS_instance_name\bin directory.

2. Execute the `startup.sh` script file.

Note:

If your Integration Server has been configured to request a master password for outbound password encryption, you will be prompted for this password in a popup window or from the server console. Refer to [“Managing Outbound Passwords” on page 538](#) for more information about this password.

Starting a Server Instance from the Command Prompt

You can start server instances from the command prompt. When you start a server instance from the command prompt, you have the option of overriding certain settings in the configuration file. You can also start the server instance in "debug" mode, so you can record or display server activity.

Note:

If you are running the Windows Server 2008 r2 operating system with the User Account Control security feature enabled, the command prompt you use to run the startup.bat service must be launched with full Administrator privileges. To launch the command prompt with full Administrator privileges, navigate to All Programs > Accessories, right-click on the Command Prompt listing, and select the Run as Administrator option. If you are not logged into the operating system with Administrator privileges, you will be prompted to supply Administrator credentials.

➤ **To start a server instance from the command prompt in Integration Server**

1. At a command prompt, type the following command:

```
cd
    Software AG_directory
    \profiles\IS_instance_name\bin
```

where *instance_name* is the name of the Integration Server instance.

Note:

The startup.bat/sh and shutdown.bat/sh scripts contained in the *Integration Server_directory* \instances*instance_name*\bin directory are deprecated. Software AG recommends the use of the scripts contained in the *Software AG_directory* \profiles\IS_instance_name\bin directory. If you will manage Integration Server through Command Central, you *must* use the scripts located in the *Software AG_directory* \profiles\IS_instance_name\bin directory.

2. Type the following command to start the server instance:

For Windows: bin\startup.bat -switch -switch ...

For UNIX: bin/startup.sh -switch-switch ...

where *switch* is optional and can be one of the following :

switch	Description
-port <i>portNumber</i>	Specifies the port on which the server listens for HTTP requests. <i>portNumber</i> specifies the TCP/IP port number Example: -port 8080 This switch overrides the value assigned to watt.server.port.

Note:

In addition to overriding the value of watt.server.port, the -port switch permanently adds a new HTTP port to the WmRoot package. This new port is added as the primary port and contains default values. If a port with the same TCP/IP number already

switch**Description**

exists in the WmRoot package, the `-port` switch overrides its settings with the new default values. In effect, deleting the existing port and then adding a new port with default settings.

Note:

To use port 80 (the standard for HTTP) or port 443 (the standard for HTTPS), UNIX users must be running as "root." For security reasons, a better method is to use a higher number port (5555 for HTTP and 8080 for HTTPS), and if necessary have the firewall remap port 80 to the desired port. See ["Architecture" on page 26](#) for a discussion of remapping ports.

`-debug level`

Specifies the level of detail you want the server to maintain in its server log for this session.

level indicates the level of detail you want to record in the log.

Specify...	To record...
Fatal	Fatal messages only.
Error	Error and fatal messages.
Warn	Warning, error, and fatal messages.
Info	Informational, warning, error, and fatal messages.
Debug	Debug, informational, warning, error, and fatal messages.
Trace	Trace, debug, informational, warning, error, and fatal messages.

For this session, this switch overrides the value specified for the Default facility on the **Settings > Logging** page and assigned to `watt.debug.level`.

Note:

Prior to Integration Server 7.1, Integration Server used a number-based system to set the level of debug information written to the server log. Integration Server maintains backward compatibility with this system. For more information about the number-based logging levels, see the description of the `watt.debug.level` property in ["Server Configuration Parameters" on page 1017](#).

switch	Description								
-log <i>destination</i>	Specifies where you want the server to write its server log information for this session. Specify one of the following for <i>destination</i> :								
	<table border="1"> <thead> <tr> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>filename</i></td> <td>Specify the fully qualified path or relative path to the file in which you want the server to write server log information for this session. Relative path is relative to the Integration Server home directory: <i>Integration Server_directory</i> <i>\instances\instance_name</i> The <i>filename</i> must specify a directory and filename. The default destination is controlled by the <code>watt.debug.logfile</code> server configuration parameter.</td> </tr> <tr> <td>none</td> <td>Write server log information to the computer screen (STDOUT). Server log messages written to STDOUT include the identifier "ISSERVER" to help differentiate server log messages from other messages written to the console.</td> </tr> <tr> <td>both</td> <td>Write server log information to the computer screen (STDOUT) and to the destination specified by the <code>watt.debug.logfile</code> parameter.</td> </tr> </tbody> </table>	Option	Description	<i>filename</i>	Specify the fully qualified path or relative path to the file in which you want the server to write server log information for this session. Relative path is relative to the Integration Server home directory: <i>Integration Server_directory</i> <i>\instances\instance_name</i> The <i>filename</i> must specify a directory and filename. The default destination is controlled by the <code>watt.debug.logfile</code> server configuration parameter.	none	Write server log information to the computer screen (STDOUT). Server log messages written to STDOUT include the identifier "ISSERVER" to help differentiate server log messages from other messages written to the console.	both	Write server log information to the computer screen (STDOUT) and to the destination specified by the <code>watt.debug.logfile</code> parameter.
Option	Description								
<i>filename</i>	Specify the fully qualified path or relative path to the file in which you want the server to write server log information for this session. Relative path is relative to the Integration Server home directory: <i>Integration Server_directory</i> <i>\instances\instance_name</i> The <i>filename</i> must specify a directory and filename. The default destination is controlled by the <code>watt.debug.logfile</code> server configuration parameter.								
none	Write server log information to the computer screen (STDOUT). Server log messages written to STDOUT include the identifier "ISSERVER" to help differentiate server log messages from other messages written to the console.								
both	Write server log information to the computer screen (STDOUT) and to the destination specified by the <code>watt.debug.logfile</code> parameter.								

The *filename* and none values override the value assigned to `watt.debug.logfile` for this session.

Note:

For SSL session logging, a -log switch value specifies where Integration Server writes the SSL session information. For a -log switch value of none, Integration Server writes the SSL session information to a separate file (inboundSSLSessions.log). For a switch value of both, Integration Server writes the SSL session information to the inboundSSLSessions.log file and to the console. For more information on SSL session logging, see ["Setting Up SSL Session Logging" on page 473](#).

Note:

switch	Description
	<p>A <code>-log</code> switch value of <code>none</code> or <code>both</code> also determines where Microservices Runtime or an Integration Server equipped with an Microservices Runtime license writes the configuration variables log. However, Microservices Runtime ignores a filename value. If you specify a filename, Microservices Runtime writes the configuration variables log file to this location only:</p> <p><i>Integration Server_directory</i> <i>/instances/instanceName/logs/configurationvariables.log</i>. For more information about configuration variable templates and the associated logging, see <i>Developing Microservices with webMethods Microservices Runtime</i>.</p>
<p><code>-quiesce</code></p>	<p>Specifies to start the server in quiesce mode.</p> <p>For more information about quiesce mode, see “Quiescing the Server for Maintenance” on page 929.</p> <p>Note: If the server cannot disable or suspend an asset or activity as it enters quiesce mode, the server displays a warning message and continues the quiesce process. If the condition stated in the warning interferes with a maintenance task you intend to perform, resolve the issue stated in the warning and then restart the server in quiesce mode.</p>

What Happens When You Start the Server?

When you start Integration Server, it performs a series of initialization steps to make itself ready for client requests. The server:

1. Establishes the operating environment by using the configuration parameters located in the configuration file (*Integration Server_directory* \instances*instance_name*\config\server.cnf).
2. Initializes processes that perform internal management.
3. Loads information about all the enabled packages and their services that reside in the *Integration Server_directory* \instances*instance_name*\packages directory. If a package depends on other packages, the server loads the prerequisite packages first. The server does not load disabled packages.
4. Executes the startup services for each loaded package.
5. Initializes the guaranteed delivery engine. The server checks the job store for pending guaranteed delivery transactions. It retries the pending transactions as the guaranteed delivery configuration settings specify. For more information, refer to [“Configuring Guaranteed Delivery” on page 739](#).
6. Schedules internal system tasks.

How to Tell if the Server Is Running Correctly

To determine whether your server is running, start your browser and point it to the Integration Server instance. (See [“Starting the Integration Server Administrator”](#) on page 78 if you need instructions for this step.)

- If the server is running, you will be prompted for a name and password.
- If the server is not running, your browser will issue an error message similar to the following:

```
"Cannot open the Internet site http://localhost:5555."  
"A connection with the server could not be established."
```

If Integration Server detects a problem with the master password or outbound passwords at startup, it will place you in *safe mode*, which is a special mode from which you can diagnose and correct problems. When Integration Server is in safe mode, it displays the Integration Server Administrator, but Integration Server is not connected to any external resources.

When you are placed into safe mode because of problems with the master password or outbound passwords, you will see the following message in the upper left corner of the Server Statistics screen of the Integration Server Administrator:

```
SERVER IS RUNNING IN SAFE MODE. Master password sanity check failed -- invalid  
master password provided.
```

These problems can be caused by a corrupted master password file, a corrupted outbound password file, or by simply mis-typing the master password when you are prompted for it. If you suspect you have mis-typed the password, shut down the server and restart it, this time entering the correct password. If this does not correct the problem, refer to [“When Problems Exist with the Master Password or Outbound Passwords at Startup”](#) on page 546 for instructions.

Running Integration Server as a Windows Application vs. a Windows Service

Integration Server can run as either a Windows application or a Windows service.

- **Use a Windows application** if you want to control when the Integration Server initializes. When Integration Server is a Windows application, you must manually start it.

If you installed the Integration Server as a Windows service and now want it to be a Windows application, you can manually remove the Windows service for the Integration Server. After removing the Windows service, the Integration Server will still be available as a Windows application. See [“Switching the Server from a Windows Service to a Windows Application”](#) on page 45.

- **Use a Windows service** to have Integration Server automatically initialize when the machine on which it is installed initializes. When you use a Windows service, you do not have to manually restart Integration Server following a machine restart.

If you installed Integration Server as a Windows application and now want it to be a Windows service, you can manually register the Integration Server service.

See “Switching the Server from a Windows Application to a Windows Service” on page 45.

Note:

If you want the Integration Server to prompt for the master password for outbound passwords at server initialization, do not run it as a Windows service. For more information about outbound passwords and the master password, refer to “Configuring Integration Server for Secure Communication” on page 457.

You can run multiple Integration Server instances as applications and multiple Integration Server instances as services on a single machine. For example, you can run two instances of Integration Server as an application and two instances of Integration Server as services on the same machine.

Note:Microservices Runtime cannot be registered as a Windows service. An Integration Server equipped with an Microservices Runtime license can be registered as a Windows service.

Switching the Server from a Windows Service to a Windows Application

If Integration Server was installed as a Windows service, and you want Integration Server to run as a Windows application, you must remove the Windows service for the Integration Server.

➤ To manually remove the Windows service for the Integration Server

1. If the Windows service is running, stop it. You can stop the Windows service by either using the Integration Server Administrator to shut down the Integration Server or from the **Services** dialog box in the Microsoft Windows Control Panel.
2. Open a command prompt, navigate to the *Software AG_directory \profiles\IS_instance_name\bin* directory and run the following command to remove the Integration Server service:

```
service.bat -remove
```

Note:

The installSvc.bat file located in *Integration Server_directory \instances\instance_name \support\win32* directory is deprecated. Software AG recommends using the service.bat file from the *Software AG_directory \profiles\IS_instance_name\bin* directory.

Switching the Server from a Windows Application to a Windows Service

To switch the server from a Windows application to a Windows service, you must manually register the Integration Server instance to run as a Windows service.

Note:

The user whose identity will be used to run the Integration Server as a Windows service must be a Windows Administrator or member of the Windows Administrators group.

> To manually register an Integration Server instance to run as a Windows service

1. Edit the `custom_wrapper.conf` file to fit your environment.

For example, you might change the Java minimum heap size by updating the `wrapper.java.initmemory` property. The `custom_wrapper.conf` file is located in the `Software AG_directory \profiles\IS_instance_name\configuration` directory.

For more information about setting values in the `custom_wrapper.conf` file, see *Software AG Infrastructure Administrator's Guide*.

2. Open a command prompt, navigate to the `Software AG_directory \profiles\IS_instance_name\bin` directory and run the following command to create the Integration Server service:

```
service.bat -install
```

Note:

The `installSvc.bat` file located in `Integration Server_directory \instances\instance_name \support\win32` directory is deprecated. Software AG recommends using the `service.bat` file from the `Software AG_directory \profiles\IS_instance_name\bin` directory.

In the Microsoft Windows Control Panel in the Services dialog box, verify that the service for Integration Server has been created.

3. Start the service from one of the following places:

- **Services** dialog box in the Microsoft Windows Control Panel
- Command prompt using the following command:

```
net start <SVCNAME>
```

where `<SVCNAME>` is the name of the service created in the previous step.

Increasing File Descriptors on Mac OS X or Other UNIX System

The ability of Integration Server to handle traffic is constrained by the number of file descriptors available to the Integration Server process. On most systems, 64 file descriptors are available to each process by default. If Integration Server is running on a Mac OS X or other UNIX system, Software AG recommends that you increase the number of file descriptors available to the Integration Server process to at least 1024.

Note:

You might have to increase this number depending on the number of files Integration Server needs to have open at one time. It is dangerous to set the `rlim_fd_max` value higher than 1024 because of limitations with the `select` function, so if Integration Server requires more file descriptors, set the `setrlimit` value directly.

Changing Settings in the Configuration File `custom_wrapper.conf`

The `custom_wrapper.conf`, which is used at Integration Server startup, contains configuration settings. You can override these settings for a single Integration Server session by starting Integration Server from the command prompt and using switches to override values in the `custom_wrapper.conf` file. However, you might want to permanently change some of the configuration values in the `custom_wrapper.conf` so that all sessions use the modified configuration values.

Note: Microservices Runtime does not use the `custom_wrapper.conf`. For Microservices Runtime, set configuration information in the following file: `Integration Server_directory/bin/setenv.bat(sh)`

➤ To override settings in the configuration file

1. In a text editor, open the `custom_wrapper.conf` file from the following directory:

`Software AG_directory \profiles\IS_instance_name\configuration`

where `instance_name` is the name of the Integration Server instance.

2. Add the following property to `custom_wrapper.conf`:

```
wrapper.app.parameter.n=switch
```

where `n` is the next unused sequential number for the `wrapper.app.parameter` properties in the file and `switch` is the switch command.

```
wrapper.app.parameter.n=switch_parameter
```

where `n` is the sequential number and `switch_parameter` is the value of the switch.

Most switches require that you add an additional property to `custom_wrapper.conf` as the very next property in the sequence, as follows:

For example, to change the default port number to 8080, you would enter the following to `custom_wrapper.conf`:

```
wrapper.app.parameter.7=-port
wrapper.app.parameter.8=8080
```

For more information about the `wrapper.app.parameter` property, see *Software AG Infrastructure Administrator's Guide*.

The following table describes the switches you can use to override settings in the configuration file:

switch	Description		
<code>-port</code>	<p>Specifies the port on which the server listens for HTTP requests. If you add a <code>wrapper.app.parameter</code> property to <code>custom_wrapper.conf</code> for the <code>-port</code> switch, you must also add a <code>wrapper.app.parameter</code> property in which you specify the TCP/IP port number to use.</p> <p>The first <code>wrapper.app.parameter</code> property specifies that you want to override the port by using the <code>-port</code> switch, and the next <code>wrapper.app.parameter</code> property specifies the port number to use. For example:</p> <pre>wrapper.app.parameter.7=-port wrapper.app.parameter.8=8080</pre> <p>Note: Keep the following points in mind when overriding the port number:</p> <ul style="list-style-type: none"> ■ This switch overrides the value assigned to <code>watt.server.port</code>. ■ The <code>-port</code> switch permanently adds a new HTTP port to the WmRoot package. This new port is added as the primary port and contains default values. If a port with the same TCP/IP number already exists in the WmRoot package, the <code>-port</code> switch overrides its settings with the new default values. In effect, deleting the existing port and then adding a new port with default settings. ■ To use port 80 (the standard for HTTP) or port 443 (the standard for HTTPS), UNIX users must be running as "root." For security reasons, a better method is to use a higher number port (5555 for HTTP and 8080 for HTTPS), and if necessary have the firewall remap port 80 to the desired port. See "Architecture" on page 26 for a discussion of remapping ports. 		
<code>-debug</code>	<p>Specifies the level of detail you want the server to maintain in its server log for this session. The <code>-debug</code> switch overrides the value specified for the Default facility on the Settings > Logging page and assigned to <code>watt.debug.level</code>.</p> <p>If you add a <code>wrapper.app.parameter</code> property to <code>custom_wrapper.conf</code> for the <code>-debug</code> switch, you must also add a <code>wrapper.app.parameter</code> property in which you specify the level of detail you want to record in the log. You can specify the following levels:</p> <table border="0"> <thead> <tr> <th style="text-align: left;"><u>Specify</u></th> <th style="text-align: left;"><u>To record</u></th> </tr> </thead> </table>	<u>Specify</u>	<u>To record</u>
<u>Specify</u>	<u>To record</u>		

switch	Description
Fatal	Fatal messages only.
Error	Error and fatal messages.
Warn	Warning, error, and fatal messages.
Info	Informational, warning, error, and fatal messages.
Debug	Debug, informational, warning, error, and fatal messages.
Trace	Trace, debug, informational, warning, error, and fatal messages.

The first `wrapper.app.parameter` property specifies that you want to override the debug level by specifying the `-debug` switch, and the next `wrapper.app.parameter` property specifies the level of detail you want to record in the log. For example:

```
wrapper.app.parameter.11=-debug
wrapper.app.parameter.12=fatal
```

Note:

Prior to Integration Server 7.1, Integration Server used a number-based system to set the level of debug information written to the server log. Integration Server maintains backward compatibility with this system. For more information about the number-based logging levels, see the description of the `watt.debug.level` property in [“Server Configuration Parameters” on page 1017](#).

<code>-log</code>	Specifies where you want the server to write its server log information for this session. If you add a <code>wrapper.app.parameter</code> property to <code>custom_wrapper.conf</code> for the <code>-log</code> switch, you must also add a <code>wrapper.app.parameter</code> property in which you specify the destination of the log information. You can specify the following destinations:
-------------------	---

<u>Option</u>	<u>Description</u>
<i>filename</i>	Specify the fully qualified path or relative path to the file in which you want the server to write server log information for this session. Relative path is relative to the Integration Server home directory:

switch	Description
	<p><i>Integration Server_directory</i> <i>\instances\instance_name</i></p> <p>The <i>filename</i> must specify a directory and filename.</p> <p>The default destination is controlled by the <code>watt.debug.logfile</code> server configuration parameter.</p>
none	Write server log information to the computer screen (STDOUT).
both	Write server log information to the computer screen (STDOUT) and to the destination specified by the <code>watt.debug.logfile</code> parameter.

The first `wrapper.app.parameter` property specifies that you want to override the destination of the log information by specifying the `-log` switch, and the next `wrapper.app.parameter` property specifies either the complete path for the home directory or none. For example:

```
wrapper.app.parameter.13=-log
wrapper.app.parameter.14=none
```

Note:

A switch value of *filename* or none overrides the value assigned to `watt.debug.logfile` for this session.

Note:

A `-log` switch value of none or both also determines where Microservices Runtime or an Integration Server equipped with an Microservices Runtime license writes the configuration variables log. However, Microservices Runtime ignores a filename value. If you specify a filename, Microservices Runtime writes the configuration variables log file to this location only:

Integration Server_directory
/instances/instanceName/logs/configurationvariables.log.

<code>-quiesce</code>	Specifies that Integration Server starts the server in quiesce mode. When you add a <code>wrapper.app.parameter</code> property to <code>custom_wrapper.conf</code> for the <code>-quiesce</code> switch, Integration Server starts in quiesce mode. The <code>-quiesce</code> switch does not require an additional <code>wrapper.app.parameter</code> property. For example:
-----------------------	--

- | switch | Description |
|---------------|--|
| | <pre>wrapper.app.parameter.15=-quiesce</pre> <p>For more information about quiesce mode, see “Quiescing the Server for Maintenance” on page 929.</p> |
- In the `custom_wrapper.conf`, update the `wrapper.app.parameter.2` property to reflect the total number of `wrapper.app.parameter` properties that you added in the previous steps.

For example, if `wrapper.app.parameter.2` is set to 4 (the default) and you add two `wrapper.app.parameter` properties, you would increase the value of `wrapper.app.parameter.2` by two. After making your edits, the `wrapper.app.parameter.2` would appear as follows:

```
wrapper.app.parameter.2=6
```

For more information about the `wrapper.app.parameter` property, see *Software AG Infrastructure Administrator's Guide*.
 - Save and close `custom_wrapper.conf`.

Passing Java System Properties to Integration Server

You can pass Java system properties to Integration Server by modifying the `custom_wrapper.conf` file.

Note: Microservices Runtime does not use the `custom_wrapper.conf`. For Microservices Runtime, set Java system properties in the following file using the `JAVA_CUSTOM_OPTS` property: `Integration Server_directory/bin/setenv.bat(sh)`

➤ To pass Java system properties to Integration Server

- Open the `custom_wrapper.conf` file in a text editor. You can find the `custom_wrapper.conf` file in the following directory:


```
Software AG_directory \profiles\IS_instance_name\configuration
```
- Add a `wrapper.java.additional.n` property that specifies the property name and value that you want to pass to Integration Server, where *n* is a unique sequence number. The property name must be preceded by `-D`.

For example, the `wrapper.java.additional` properties in the `custom_wrapper.conf` file would look similar to the following:

```
wrapper.java.additional.11=-Dmy.prop1=value1
wrapper.java.additional.12=-Dmy.prop2=value2
```

For more information about setting values in the `custom_wrapper.conf` file, see *Software AG Infrastructure Administrator's Guide*.

3. Save and close the file.
4. Restart the server for the changes to take effect.

Shutting Down the Integration Server

You can shut down the server to stop the Integration Server and all active sessions. You can perform this task from Integration Server Administrator or from the command prompt.

For information about shutting down Microservices Runtime, see *Developing Microservices with webMethods Microservices Runtime*.

Shutting Down the Integration Server from Integration Server Administrator

Use this procedure to shut down the Integration Server and all active sessions from Integration Server Administrator.

> To shut down the server

1. Open the Integration Server Administrator if it is not already open.
2. In the upper right corner of any Integration Server Administrator screen, click **Shutdown and Restart**.
3. Select whether you want the server to wait before shutting down or to shut down immediately.

Delay number minutes or until all client sessions are complete. Specify the number of minutes you want the Integration Server to wait before shutting down. It then begins monitoring user activity and automatically shuts down when all non-administrator sessions complete or when the time you specify elapses (whichever comes first).

Perform action immediately. The server and all active sessions terminate immediately.

4. For instructions on how to view the active sessions, refer to [“Viewing Active Sessions” on page 54](#).
5. Click **Shutdown**.

Shutting Down Integration Server from Windows

You can shut down any instance of Integration Server and all active session from the Windows start menu.

Note:

If you are running the Windows Server 2008 r2 operating system or Windows Server 2012 with the User Account Control security feature enabled, you must shut down Integration Server with full Administrator privileges. To shut down Integration Server with full Administrator privileges, navigate to the list of programs or applications for **Software AG**. For example, in Windows Server 2008, navigate to **All Programs > Software AG**. Right-click **Stop *instanceName***, and select the **Run as Administrator** option. If you are not logged into the operating system with Administrator privileges, you will be prompted to supply Administrator credentials.

➤ To shut down Integration Server on Windows

1. Click **Start**.
2. In the **All Programs** menu, click the **Software AG** folder.
3. Click the **Stop Servers** folder.
4. Click **Stop Integration Server**.
5. Click **Stop Integration Server *instanceName***.

Shutting Down Integration Server from the Command Prompt

Use this procedure to shut down the server and all active sessions from the command prompt.

➤ To shut down the server from the command prompt

1. At a command prompt, type the following command to switch to the server instance's home directory:

```
cd
    Software AG_directory
    \profiles\IS_instance_name\bin
```

where *instance_name* is the name of the Integration Server instance.

Note:

The startup.bat/sh and shutdown.bat/sh scripts contained in the *Integration Server_directory* \instances*instance_name*\bin directory are deprecated. Software AG recommends the use of the scripts contained in the *Software AG_directory* \profiles\IS_*instance_name*\bin directory. If you will manage Integration Server through Command Central, you *must* use the scripts located in the *Software AG_directory* \profiles\IS_*instance_name*\bin directory.

2. Type the following command to stop the server:

For Windows: shutdown.bat

For UNIX: `shutdown.sh`

Viewing Active Sessions

When Integration Server is running, you can view the sessions that are currently active.

> To view active sessions

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Statistics**.
3. Click on the current number of sessions.


Killing Sessions

You can kill a single session or all sessions except the session you are currently running.

> To kill sessions


1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Statistics**.
3. In the **Total Sessions** field, in the **Current** column, click the current number of sessions.

Integration Server displays the **Sessions** screen.

4. Click  in the **Kill** column of the session you want to kill.

Note:

Alternatively, you can click **Kill All Except Your Session** above **Current Sessions** to kill all sessions except the session you are currently running.

Integration Server kills the current sessions that are registered in the Integration Server database and not the in-flight sessions. Integration Server does not kill active JMS trigger sessions. The icon,  in the **Kill** column is disabled for active JMS trigger sessions until the sessions expire.

Viewing the Integration Server Process ID

Each instance of Integration Server runs on an operating system as a separate JVM process. There are times when it is necessary to perform a thread dump or stop a particular instance of Integration Server (for example, in Windows Task Manager). In order to do this, it is necessary to know the process ID of the Integration Server process (or *server process ID*) on which to perform the

particular operation. You can view the server process ID in the Integration Server Administrator About page.

➤ **To view the process ID**

1. Open the Integration Server Administrator if it is not already open.
2. Click **About** in the upper right-hand part of the screen.
3. Scroll down to the **Server Process Id** field in the **Server Environment** area.

Restarting the Integration Server

Restart the server when you need to stop and reload Integration Server. You should restart the server when:

- **You make certain configuration changes.** Some configuration changes require the server to be restarted before they take effect. This document indicates when you are required to restart the server for configuration changes.
- **You want to incorporate updated services that cannot be dynamically reloaded.** This typically occurs for non-Java services.

➤ **To restart the server**

1. Open the Integration Server Administrator if it is not already open.
2. In the upper right corner of any Integration Server Administrator screen, click **Shutdown and Restart**.
3. Select whether you want the server to wait before restarting or to restart immediately.
 - **Delay number minutes or until all client sessions are complete.** Specify the number of minutes you want the Integration Server to wait before restarting. It then begins monitoring user activity and automatically restarts when all non-administrator sessions complete or when the time you specify elapses (whichever comes first).
 - **Perform action immediately.** The server and all active sessions terminate immediately. Then the server restarts.

For instructions on how to view the active sessions, refer to [“Viewing Active Sessions” on page 54](#).

4. Click **Restart**.

Server Recovery

If a hardware or software problem causes Integration Server to fail, restart the server using the normal startup procedure. The server will attempt to perform cleanup and initialization processes to reset the operating environment.

As part of the recovery process, the server automatically:

- Reloads the cache environment to its pre-failure state.
- Restores the transaction manager's guaranteed delivery queues. See [“Configuring Guaranteed Delivery” on page 739](#) for additional information about guaranteed delivery recovery options.

Services that your site has created might have their own unique recovery requirements. Consult with your developers for information about these requirements.

Some circumstances might require manual intervention to restart the server.

Tip:

Before restarting Integration Server, you can collect diagnostic data to troubleshoot run-time issues. For information about using the diagnostic port and utility, see [“Diagnosing the Integration Server” on page 937](#). Also refer to this chapter for information on generating thread dump to troubleshoot reasons for server slowdown or unresponsiveness.

Unapplied Changes to Integration Server Configuration Files

If Integration Server shut down improperly while you were making changes that affect configuration files (*.cnf), Integration Server might display the following message upon restart:

```
[ISS.0025.0070C] Integration Server detected files in config/work directory which suggests unapplied configuration changes.
```

When saving changes to configuration files, Integration Server first saves the configuration changes in a temporary file in the *Integration Server_directory/instances/instanceName/config/work* directory. After saving the changes in a temporary file, Integration Server moves the temporary file to the actual configuration file. This ensures that if unexpected behavior occurs before the configuration changes are initially saved to the temporary file, only the temporary file is impacted. The actual configuration file is not corrupted. At start up, if Integration Server detects files in the *Integration Server_directory/instances/instanceName/config/work*, it suggests that changes to a configuration file were not saved to the temporary file and that, therefore, the changes were not made to the actual configuration file. Use the contents of the *Integration Server_directory/instances/instanceName/config/work* directory to determine which configuration files were in the process of being changed. Decide whether or not you want to redo the changes to the configuration files. Delete the files under the directory and redo the configuration change if you so choose.

Integration Server Data Integrity and Recoverability Considerations

The Integration Server utilizes a webMethods physical storage technology to persist critical operational data. This storage technology employs database-like technology of logging and transactional management. Under normal operations these facilities maintain the integrity, consistency, and recoverability of data persisted to these files. However, even with these safeguards, abnormal server shutdown and catastrophic failures can occur, which could result in these files being left in an unrecoverable state.

Shutting down Integration Server by any means other than the Integration Server Administrator *may* result in critical data files being left in an unrecoverable state. This will result in the inability to restart the Integration Server without manual intervention to remove or recover the damaged data files.

Important:

Establish site-specific backup and restore procedures to protect these critical data files.

The mission-critical nature of the data stored in the Integration Server's data files requires that it be backed up periodically for disaster recovery. As in all critical data resources, the potential exists for a physical failure to leave the Integration Server data files in a corrupted state. In these situations the method of recovery is to replace these data files with the most current backup. The frequency and nature of these backups depends on the critical nature of the data being stored. Backups of these data files should be an offline process with the Integration Server in an idle or shutdown state, i.e. no disk activity.

Important:

Implement site-specific procedures to periodically backup the critical Integration Server data files. You can use any file-system backup utility. Perform the backup process only when the Integration Server is shutdown or in a quiesce state, (no disk activity). This restriction ensures that the backup will capture these critical data files in a consistent state. Backing up an active Integration Server may result in capturing a snapshot of the data files that are in an inconsistent state and therefore unusable for recovery purposes.

Critical Integration Server Data Files

There are two subdirectories in Integration Server's currently working directory that contain critical data files that must be backed up for recovery purposes. These subdirectories are:

- `./DocumentStore`

The files in this subdirectory contain the locally persisted documents being processed by Integration Server. The loss of these files will result in the loss of any persisted documents. Back up these six files:

`ISResubmitStoredata0000000`

`ISResubmitStorelog0000000`

`ISTransStoredata0000000`

ISTransStorelog0000000

TriggerStoredata0000000

TriggerStorelog0000000

■ `./WmRepository4`

The files in this subdirectory contain metadata for Integration Server. The loss of these files could result in loss of configuration information and may require manual reconfiguration.

Back up these two files:

FSDdata0000000

FSDlog0000000

The Java Service Wrapper

webMethods Integration Server runs on the on the Software AG Common Platform, which in turn runs in a Java Virtual Machine (JVM). The Java Service Wrapper is an application developed by Tanuki Software, Ltd. It is a utility program that launches the JVM in which Integration Server runs.

In addition to launching the JVM, the Java Service Wrapper offers features for monitoring the JVM, logging console output, and generating thread dumps. The following sections describe how Integration Server uses the features of the Java Service Wrapper. For an overview of the Java Service Wrapper, see the webMethods cross-product document, *Software AG Infrastructure Administrator's Guide*.

Note:

Microservices Runtime does not use the Java Service Wrapper.

The Java Service Wrapper Configuration Files

For Integration Server, the configuration files for the Java Service Wrapper reside in the following directory.

`Software AG_directory \profiles\IS_instanceName\configuration`

When you start Integration Server, property settings in the following files determine the configuration of the JVM and the behavior of the logging and monitoring features of the Java Service Wrapper.

File name

Description

wrapper.conf

Contains property settings that are installed by Integration Server.

Important:

File name	Description
	Do not modify the contents of this file unless asked to do so by Software AG.
custom_wrapper.conf	<p>Contains properties that modify the installed settings in wrapper.conf.</p> <p>If you need to modify the property settings for the Java Service Wrapper, you make your changes in this file.</p>
	<p>Note: Beginning with Integration Server version 9.7, Integration Server no longer obtains settings from setenv.bat/sh or server.bat/sh. At startup, Integration Server obtains all classpath modifications from custom_wrapper.conf. For more information about how Integration Server builds classpaths, see “How the Integration Server Classpath Is Set” on page 914.</p>

The following sections describe configuration changes that Integration Server supports for Java Service Wrapper. For Integration Server, do not make any configuration changes to the Java Service Wrapper other than the ones described in the following sections.

JVM Configuration

When the Java Service Wrapper launches the JVM, it provides configuration settings that, among other things, specify memory settings and the directories in the classpath. When you start Integration Server, the Java Service Wrapper receives these configuration settings from the wrapper.conf and custom_wrapper.conf files in the *Software AG_directory* \profiles\IS_instance_name\configuration folder.

If you need to modify the default property settings for Integration Server, you can override the settings using the custom_wrapper.conf file, which is located in the *Software AG_directory* \profiles\IS_instance_name\configuration. For more information for configuring custom_wrapper.conf properties for Integration Server, see [“Configuring the Server” on page 105](#). For general information about the Java Service Wrapper, see *Software AG Infrastructure Administrator’s Guide*.

The Wrapper Log

The Java Service Wrapper records console output in a log file. The log contains the output sent to the console by the wrapper itself and by the JVM in which Integration Server is running. The wrapper log is especially useful when you run Integration Server as a Windows service, because console output is normally not available to you in this mode.

The Java Service Wrapper log for Integration Server is located in the following file:

Software AG_directory \profiles\IS_instanceName\logs\wrapper.log

To view the log, open the log file in a text editor.

Logging Properties

The `wrapper.console` and `wrapper.log` properties in the wrapper configuration files determine the content, format, and behavior of the wrapper log.

The logging settings that Integration Server installs are suitable for most environments. However, you can modify the following properties if the installed settings do not suit your needs. For procedures and additional information, see the webMethods cross-product document, *Software AG Infrastructure Administrator's Guide*.

Property	Value
<code>wrapper.console.format</code>	The format of the messages displayed in the console.
<code>wrapper.console.loglevel</code>	The level of detail displayed in the console.
<code>wrapper.logfile</code>	The file into which console output for Integration Server is logged.
<code>wrapper.logfile.format</code>	The format of the messages recorded in the wrapper log file.
<code>wrapper.logfile.loglevel</code>	The level of detail recorded in the wrapper log file. This setting must be a level that is equal to or lower than the setting in <code>wrapper.console.loglevel</code> .
<code>wrapper.logfile.maxsize</code>	The maximum size to which the log can grow.
<code>wrapper.logfile.maxfiles</code>	The number of old logs the Java Service Wrapper retains.
<code>wrapper.syslog.loglevel</code>	Which messages are written to the Event Log on Windows systems, or the syslog on UNIX systems.

Fault Monitoring

The Java Service Wrapper can monitor the JVM for the certain conditions and then restart the JVM or perform other actions when it detects these conditions.

The following table describes the fault-monitoring features Integration Server uses or allows you to configure. To learn more about these features, see the webMethods cross-product document, *Software AG Infrastructure Administrator's Guide*.

Feature	Enabled?	User configurable?
JVM timeout	Yes	No. Do not modify the <code>wrapper.ping</code> properties unless asked to do so by Software AG.
Deadlock detection	No	No. Do not enable this feature.
Console filtering	No	No. Do not enable this feature.

Generating a Thread Dump

The Java Service Wrapper provides a utility for generating a thread dump of the JVM. A thread dump can help you locate thread contention issues that can cause thread blocks or deadlocks.

For information about generating a thread dump using the Java Wrapper Service, see the webMethods cross-product document, *Software AG Infrastructure Administrator's Guide*.

Re-registering a Windows Service after Installing a New Version of the Java Service Wrapper

If you are running Integration Server as a Windows service, you must re-register the service when the version of the Java Service Wrapper changes. Software AG may provide updated versions of the Java Service Wrapper in a webMethods Shared Libraries Tanuki fix or in a new release. Re-registering the service involves removing the Windows service and then registering (installing) it again.

➤ To re-register the Windows service for Integration Server

1. If the Windows service is running, stop it. You can stop the Windows service by either using the Integration Server Administrator to shut down the Integration Server or from the **Services** dialog box in the Microsoft Windows Control Panel.
2. Open a command prompt, navigate to the *Software AG_directory* \profiles\IS_instance_name\bin directory and run the following command to remove the Integration Server service:

```
service.bat -remove
```

3. Run the following command to register the Windows service for Integration Server:

```
service.bat -install
```


4 Running Multiple Integration Server Instances

■ Overview of Integration Server Instances	64
■ Guidelines for Running Multiple Integration Server Instances on the Same Machine	64
■ About Creating a New Integration Server Instance	64
■ About the is_instance Script	65
■ Creating a New Integration Server Instance	66
■ Updating an Integration Server Instance	69
■ Updating Packages on a Server Instance	70
■ Updating Database Properties of a Server Instance	71
■ Deleting Packages from a Server Instance	72
■ Updating Language Packs on a Server Instance	73
■ Deleting a Server Instance	74

Overview of Integration Server Instances

You can create and run multiple instances of Integration Server on the same host machine with a single physical installation of webMethods Integration Server. This is not the same as clustering, which is described in *webMethods Integration Server Clustering Guide*. When running multiple instances of Integration Server on the same host machine, each server has its own packages, configuration files, log files, and updates; however, unlike clustering, all of the server instances can share common libraries. Sharing the common library files reduces the amount of overhead and effort involved when installing fixes and patches on each server.

You manage each server instance separately using the Integration Server Administrator. You can also perform certain management, configuration, and monitoring tasks from a single location using Software AG Command Central. For information about using Software AG Command Central to manage Integration Server instances, see the Command Central documentation.

Note:Microservices Runtime does not support running multiple instances.

Guidelines for Running Multiple Integration Server Instances on the Same Machine

The following guidelines apply to running two or more Integration Server instances on the same machine:

- The common libraries are located under `SoftwareAG\common` and `Integration Server_directory\lib`. The files in these libraries can be used by all server instances. For example, you can make custom jar files available to all server instances by placing them in the `Integration Server_directory\lib\jars\custom` directory.
- Updates and custom jars located at the instance level take precedence over the updates and custom jars located in the common libraries.
- Every port number on each server instance must be unique unless the host machine has more than one NIC. The ports include the primary port, diagnostic port, and all other ports described in “[Configuring Ports](#)” on page 149.
- You do not need an additional license to create and run multiple Integration Server instances. However, creating and running additional server instances will increase the number of concurrent processes. Before creating additional server instances, check your system resources and your Software AG license.

About Creating a New Integration Server Instance

You can create a new Integration Server instance using the Integration Server instance creation script, `is_instance.bat/sh`, which is located in the `Integration Server_directory\instances` directory.

When you create a new Integration Server instance, the script creates a new folder under the `SoftwareAG\profiles\IS_instance_name` directory containing OSGI component files.

The script also creates a *home* directory for the instance under *Integration Server_directory* \instances\instance_name. The home directory contains the predefined packages described in [“Predefined Packages” on page 654](#).

If you want the server instance to use components or services contained in other packages, you must add the packages to the Integration Server. For example, if you want the server instance to use webMethods Deployer, webMethods API Gateway, or webMethods Application Platform, you must add the WmDeployer, WmAPIGateway, or WmAppPlat packages respectively to the Integration Server instance. The Software AG Installer installs packages in the package repository, which is located under the *Software AG_directory* \IntegrationServer\packages directory.

For information about how to create a new Integration Server instance, see [“Creating a New Integration Server Instance” on page 66](#). For information about how to add packages to a server instance, see [“Updating Packages on a Server Instance” on page 70](#).

About the is_instance Script

The is_instance.bat/sh script creates, updates, and deletes an Integration Server instance. It also installs, updates, or deletes packages and language packs on an Integration Server instance and updates the properties of the database, which is used by the Integration Server instance.

The is_instance.bat/sh script is located in the following directory:

Integration Server_directory \instances

Syntax

At the command prompt, use the following syntax:

Integration Server_directory \instances\is_instance.bat *command parameters*

If the command syntax is not correct, the script writes error information to the console.

is_instance Script Commands

The following table provides descriptions for the commands that can be used with the is_instance.bat/sh script.

Command	Description
create	Creates a new instance of Integration Server. For more information about using this parameter, see “Creating a New Integration Server Instance” on page 66 .
update	Adds the specified list of packages to the Integration Server instance and updates the properties of the database, which is used by the Integration Server instance. For more information about using this parameter, see “Updating

Command	Description
	Packages on a Server Instance ” on page 70 and Updating Database Properties of a Server Instance ” on page 71.
deletePackages	Deletes packages from the Integration Server instance. For more information about using this parameter, see “Deleting Packages from a Server Instance” on page 72.
updateLanguagePack	Updates language pack files for the Integration Server instance. “Deleting Packages from a Server Instance” on page 72.
delete	Deletes the Integration Server instance. For more information about using this parameter, see “Deleting a Server Instance” on page 74.

Creating a New Integration Server Instance

Use the create command to create a new Integration Server instance.

Before creating a new instance, review the information in [“Guidelines for Running Multiple Integration Server Instances on the Same Machine”](#) on page 64 and [“About Creating a New Integration Server Instance”](#) on page 64.

➤ To create a new Integration Server instance

1. Navigate to the following directory:

```
Integration Server_directory \instances
```

2. Run the instance script as follows:

```
is_instance.bat/sh create
-Dinstance.name=instance_name
-Dprimary.port=primary_port_number
-Dinstance.ip=default_bind_address
-Ddiagnostic.port=diagnostic_port_number
-Djmx.port=jmx_port_number
-Dlicense.file=license_file_location
-Dinstall.service=install_service
-Ddb.alias=database_alias_name
-Ddb.type=database_type
-Ddb.username=database_username
-Ddb.password=database_password
-Ddb.url=database_URL
-Dpackage.list=package_list
```

Where	Specify
<i>instance_name</i>	A unique name for the new instance.

Where	Specify
<i>primary_port_number</i>	Optional. A default HTTP port number for the new instance. The port number must be unique for each Integration Server instance. The default value is 5555.
<i>default_bind_address</i>	Optional. Default IP address to which to bind ports on this instance of Integration Server. Specify a bind address if your machine has multiple IP addresses and you want to use the same port number across different instances of Integration Server on the machine. In this case, each instance must specify a different bind address. If you do not specify a default bind address, the port listens to all network interfaces which prevents the same port number from being used on multiple instances. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Integration Server uses the supplied <i>default_bind_address</i> value as the value of the <code>watt.server.inetAddress</code> configuration property.</p> </div>
<i>diagnostic_port_number</i>	Optional. A default diagnostic port number for the new instance. The port number must be unique for each Integration Server instance. The default value is 9999.
<i>jmx_port_number</i>	Optional. A default JMX port number for the new instance. The port number must be unique for each Integration Server instance. The default value is 8075.
<i>license_file_location</i>	Optional. The location of your Integration Server license file. <p>The script copies the license file from the specified location to <code>IntegrationServer/instances/instance_name/config/licenseKey.xml</code> file.</p> <p>This parameter is optional; however, if a license file location is not specified, the Integration Server instance will shut down after 30 minutes.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: If the location name contains spaces, you must surround the name of the location with quotation marks (" "). For example:</p> <pre>-Dlicense.file="license file location"</pre> </div>
<i>install_service</i>	Optional. Whether to register the service for the instance. The default is <code>false</code> . <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note:</p> </div>

Where	Specify
	This option is valid only for Windows platforms.
<i>database_alias_name</i>	Optional. The alias name for the database. The default value for embedded database is Embedded Database Pool and JDBC_POOL_ALIAS for external database.
<i>database_type</i>	<p>Optional. The type of database. Specify one of the following databases:</p> <ul style="list-style-type: none"> ■ sqlserver - Microsoft SQL Server ■ oracle - Oracle ■ db2 - DB2 <p>These values are not case-sensitive. If this parameter is left unspecified, the new Integration Server instance uses the embedded IS internal database, which is the default.</p>
<i>database_username</i>	Optional. User name to connect to the database.
<i>database_password</i>	Optional. Password of the database user.
<i>database_URL</i>	<p>Optional. The connection URL for the database.</p> <p>For example:</p> <pre>jdbc:wm:oracle://<server>:<1521 port>;serviceName=<value>[;<option>=<value>...]</pre>
<i>package_list</i>	<p>Optional. A comma-separated list of packages to add to the server instance. For example, WmPRT, WmTN.</p> <p>The packages that you specify must be part of the webMethods Integration Server package repository. Software AG Installer installs packages in the package repository, which is located under the <i>Software AG_directory \IntegrationServer \packages</i> directory.</p> <p>Specify <code>all</code> to add all non-predefined packages in the webMethods Integration Server package repository located in the <i>Integration Server_directory \packages</i> directory. For a list of predefined packages, see “Predefined Packages” on page 654.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: If you want the server instance to use components or services contained in other packages, you must add the packages to the Integration Server instance. For example, if you want the server instance to use</p> </div>

Where**Specify**

webMethods Deployer, webMethods API Gateway, or webMethods Application Platform, you must add the WmDeployer, WmAPIGateway, or WmAppPlat packages respectively to the Integration Server instance.

3. Optionally, add packages to the server instance after it is created. For instructions, see [“Updating Packages on a Server Instance” on page 70](#).
4. Start the Integration Server instance by running the startup.bat/sh file located in the `Software AG_directory \profiles\IS_instance_name\bin` directory.

Note:

The startup.bat/sh and shutdown.bat/sh scripts contained in the `Integration Server_directory \instances\instance_name\bin` directory are deprecated. Software AG recommends the use of the scripts contained in the `Software AG_directory \profiles\IS_instance_name\bin` directory. If you will manage Integration Server through Command Central, you *must* use the scripts located in the `Software AG_directory \profiles\IS_instance_name\bin` directory.

Note:

If you added the package WmAppPlat, which is for webMethods Application Platform, the new Integration Server instance includes a configured instance of Tomcat. This Tomcat instance uses ports 8072 and 8074 as the default HTTP and HTTPS ports, respectively. These default ports conflict with the ports used by Tomcat on the default instance of Integration Server. You must use Command Central to change the default HTTP and HTTPS port numbers for Tomcat on the new instance of Integration Server. For more information about creating an Integration Server instance with webMethods Application Platform, see the *webMethods Application Platform User's Guide*

Updating an Integration Server Instance

Use the update command to update existing packages, to install other packages on a server instance. Also, to update the database properties when Integration Server is in the offline mode.

➤ To update an Integration Server instance

1. Run the is_instance script as follows:

```
is_instance.bat update -Dinstance.name=instance_name
-Ddb.alias=database_alias_name
-Ddb.type=database_type
-Ddb.username=database_username
-Ddb.password=database_password
-Ddb.url=database_URL
-Dpackage.list=package_list
```

For details, see [“Updating Packages on a Server Instance”](#) on page 70 and [“Updating Database Properties of a Server Instance”](#) on page 71.

Note:

When you install or update packages using the installer, you have the option to install packages on the initial instance created by the installer. If you select this option, packages are copied to that instance automatically during installation; therefore, you do not need to run the update command on that instance.

Updating Packages on a Server Instance

Use the update command to update existing packages and to install other packages on a server instance.

» To update packages on an Integration Server instance

1. Stop the Integration Server instance.
2. Navigate to the following directory:

Integration Server_directory \instances

3. Run the `is_instance` script as follows:

```
is_instance.bat update -Dinstance.name=instance_name
-Dpackage.list=package_list
```

Where**Specify**

instance_name

The name of the instance to which you want to add packages.

package_list

A comma-separated list of packages to add to the server instance. For example, `WmPRT,WmTN`.

The packages that you specify must be part of the webMethods Integration Server package repository. Software AG Installer installs packages in the package repository, which is located under the *Software AG_directory* \IntegrationServer\packages directory.

Specify `all` to add all non-predefined packages in the webMethods Integration Server package repository located in the *Integration Server_directory* \packages directory. For a list of predefined packages, see [“Predefined Packages”](#) on page 654.

Note:

If you want the server instance to use components or services contained in other packages, you must add the packages to

Where**Specify**

the instance. For example, if you want the server instance to use webMethods Deployer, webMethods API Gateway, or webMethods Application Platform, you must add the WmDeployer, WmAPIGateway, or WmAppPlat packages respectively to the instance.

The script adds the packages to the *Integration Server_directory* \instances*instance_name*\packages directory.

Updating Database Properties of a Server Instance

Use the update command to update the database properties when Integration Server is in the offline mode.

Note:

If the Integration Server instance is using embedded database, you cannot update the database properties.

➤ To update database properties of an Integration Server instance

1. Stop the Integration Server instance.
2. Navigate to the following directory:

Integration Server_directory \instances

3. Run the *is_instance* script as follows:

```
is_instance.bat update -Dinstance.name=instance_name
-Ddb.alias=database_alias_name
-Ddb.type=database_type
-Ddb.username=database_username
-Ddb.password=database_password
-Ddb.url=database_URL
```

Where**Specify**

instance_name

The name of the instance for which you want to update the database properties.

database_alias_name

The alias name for the database.

database_type

Optional. The type of database.

Specify one of the following databases:

- *sqlserver* - Microsoft SQL Server

Where	Specify
	<ul style="list-style-type: none">■ <i>oracle</i> - Oracle■ <i>db2</i> - DB2
	These values are not case-sensitive.
<i>database_username</i>	Optional. User name to connect to the database. This parameter is mandatory only if you specify the <i>database_type</i> .
<i>database_password</i>	Optional. Password of the database user. This parameter is mandatory only if you specify the <i>database_type</i> .
<i>database_URL</i>	Optional. The connection URL for the database. This parameter is mandatory only if you specify the <i>database_type</i> .

Note:

On restart, if the Integration Server instance fails to connect to the database due to incorrect database configuration, Integration Server falls back to the previous database configuration.

Deleting Packages from a Server Instance

The `deletePackages` command removes the specified package or list of packages from a server instance.

➤ To delete packages from an Integration Server instance

1. Stop the Integration Server instance.
2. Navigate to the following directory:

Integration Server_directory \instances

3. Run the `is_instance` script as follows:

```
is_instance.bat deletePackages -Dinstance.name=instance_name -Dpackage.list=package_list
```

Where	Specify
<i>instance_name</i>	The name of the instance from which you want to delete packages.
<i>package_list</i>	A comma-separated list of packages to remove from the Integration Server instance. For example, <code>packageA,packageB</code> .

The `is_instance` script removes the specified packages located under the *Integration Server_directory* \instances*instance_name*\packages*package_name* directory.

Updating Language Packs on a Server Instance

After installing, modifying, or removing language pack files on webMethods Integration Server, you must run the `updateLanguagePack` command on each server instance to synchronize and propagate the files.

Note:

When you install or update language packs using the installer, you have the option to install packages on the instance created by installer. If you select this option, language packs are synchronized on that instance automatically during installation; therefore, you do not need to run the `updateLanguagePack` command on the instance created by installer.

➤ To update language pack files on an Integration Server instance

1. Stop the Integration Server instance.
2. Navigate to the following directory:

Integration Server_directory \instances

3. Run the `is_instance` script as follows:

```
is_instance.bat updateLanguagePack -Dinstance.name=instance_name -Dinstall.mode=mode -Dpackage
```

Where	Specify
<i>instance_name</i>	The name of the instance to update.
<i>mode</i>	<p>The processing mode to indicate whether the script is to add or remove language pack files (such as jars and DSPs, HTMLs, class) from the instance. Specify one of the following:</p> <ul style="list-style-type: none"> ■ <code>install</code> to add or update language pack files. When this option is specified, the script copies the language pack files from the package repository to the specified instance.

Note:

Be sure to install the language pack using the Software AG Installer before you update the instance. If the files do not exist in the package repository, the script cannot update the instance.

- `uninstall` to remove language pack files. When this option is specified, the script removes the language pack files from the instance.

Note:

Where**Specify**

Be sure to uninstall the language pack using the Software AG Installer before running `-Dinstall.mode=uninstall` or the files will not be removed from the instance.

package_list

Optional. A comma-separated list of packages to which to add the language pack.

Note:

If you do not specify `-Dpackage.list`, you must specify `-Dfeature.list`. However, you can specify both `-Dpackage.list` and `-Dfeature.list`. You need to specify `-Dpackage.list` and/or `-Dfeature.list` only when `-Dinstall.mode` is set to `install`.

For example, to install the language pack on the Integration Server instance that functions as the API Gateway, specify:

```
-Dpackage.list=WmAPIGateway
```

feature_list

Optional. A comma-separated list of features to which to add the language pack. Review the product documentation for a list of features.

Note:

If you do not specify `-Dfeature.list`, you must specify `-Dpackage.list`. However, you can specify both `-Dpackage.list` and `-Dfeature.list`. You need to specify `-Dpackage.list` and/or `-Dfeature.list` only when `-Dinstall.mode` is set to `install`.

Deleting a Server Instance

The `delete` command removes all files and folders located under the *Integration Server_directory* \instances*instance_name* and SoftwareAG\profiles\IS_*instance_name* directories.

Important:

After you delete a server instance, you cannot retrieve any of its files or directories, including any custom directories.

> To delete an Integration Server instance

1. Stop the Integration Server instance.
2. Navigate to the following directory:

Integration Server_directory \instances

3. Run the `is_instance` script as follows:

```
is_instance.bat delete -Dinstance.name=instance_name
```

Where

instance_name

Specify

The name of the instance you want to delete.

5 Using the Integration Server Administrator

■ What Is the Integration Server Administrator?	78
■ Starting the Integration Server Administrator	78
■ Starting Integration Server Administrator on Windows	79
■ Accessing Integration Server Administrator through My webMethods	79
■ Basic Operations of Integration Server Administrator	80
■ The Server Configuration File	81
■ Software AG Command Central	81

What Is the Integration Server Administrator?

The Integration Server Administrator is an HTML-based utility you use to administer the webMethods Integration Server. It allows you to monitor server activity, manage user accounts, make performance adjustments, and set operating parameters.

You can run the Integration Server Administrator from any browser-equipped workstation on your network. (The Integration Server Administrator is a browser-based application that uses services to accomplish its work.)

Starting the Integration Server Administrator

To use the Integration Server Administrator, open your browser and point it to the port on the host machine where the Integration Server instance is running.

The server must be running in order to use Integration Server Administrator. If the server is not running, your browser will issue an error similar to the following:

```
"Cannot open the Internet site http://localhost:5555."  
"A connection with the server could not be established."
```

➤ To start the Integration Server Administrator

1. Start your browser.
2. Point your browser to the host and port where the Integration Server instance is running.

Examples

If the server were running on the default port on the same machine where you are running the Integration Server Administrator, you would type:

```
http://localhost:5555
```

If the server were running on port 4040 on a machine called QUICKSILVER, you would type:

```
http://QUICKSILVER:4040
```

3. Log on to the server with a user name and password that has administrator privileges.

If you just installed the Integration Server, you can use the following default values:

User Name: Administrator

Password: manage

Important:

Use the *exact combination* of upper- and lowercase characters shown above; user names and passwords are case sensitive.

Starting Integration Server Administrator on Windows

Start Integration Server Administrator from the Windows Start menu as described below.

➤ **To start Integration Server Administrator on Windows**

1. Click **Start**.
2. In the **All Programs** menu, click the **Software AG** folder.
3. Click the **Administration** folder.
4. Click **Integration Server Administrator**.
5. Click **Integration Server Administrator** for *instanceName* .
6. Log on to the server with a user name and password that has administrator privileges.

If you just installed the Integration Server, you can use the following default values:

User Name: Administrator

Password: manage

Important:

Use the *exact combination* of upper- and lowercase characters shown above; user names and passwords are case sensitive.

Accessing Integration Server Administrator through My webMethods

If you are working in My webMethods, you can also access Integration Server Administrator through the My webMethods ESB Administration window. Use the following procedure to add an Integration Server Administrator to the ESB Administration window.

➤ **To make Integration Server Administrator available through My webMethods**

1. From My webMethods, go to **Navigate > Applications > Administration > Integration > ESB**.
2. Click **Add Server ...** and complete the fields as follows:

Field	Entry
Description	Brief description for the Integration Server.
Server Host	Host machine for an Integration Server you want to administer from My webMethods.
Server Port	Integration Server port to which My webMethods Server will send requests.
Single Sign On	If you use the My webMethods Server single sign-on feature, users who log on to My webMethods Server will be able to administer the Integration Server without having to log on to it as well. For instructions on setting up the single sign-on feature, see “Accessing Integration Server Data through My webMethods ” on page 525.
Use Secure Connection	Whether My webMethods Server should use SSL to connect to the Integration Server.

Note:

You can only select this option if the Integration Server is configured to use SSL and certificates. For more information, see [“Configuring Integration Server for Secure Communication”](#) on page 457.

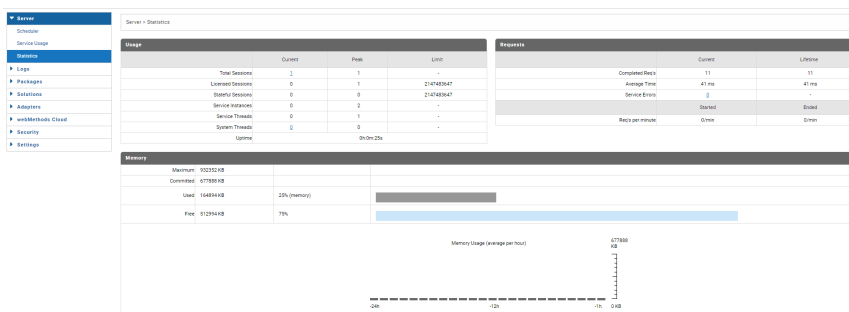
Color Choose the color to use for the border around the interface. If you have multiple s, you can use this feature to distinguish Integration Servers from each other; for example, you could use orange for Integration Servers in your production environment, and blue for Integration Servers in your test environment.

3. Click **OK**.

Basic Operations of Integration Server Administrator

When you start the Integration Server Administrator, your browser displays the **Statistics** screen.

The Integration Server Administrator Screen



The Navigation panel on the left side of the screen displays the names of menus from which you can select a task. To start a task, click a task name in the Navigation panel. The server displays a screen that corresponds to the task you select.

Logging Off the Integration Server Administrator

Log off the Integration Server Administrator when you no longer need to continue your current session. When you log off the Integration Server Administrator, the server cleans up your session and clears the cache in the browser.

➤ To log off the Integration Server Administrator

1. Click **Log Off** in the upper right corner of the Integration Server Administrator screen.

Integration Server displays a dialog box to ensure you want to log off.

2. Click **OK** to log off the Integration Server Administrator.

The browser displays a screen confirming that the session is terminated.

Getting Help

You can obtain information about the Integration Server Administrator by clicking the **Help** link in the upper right corner of any Integration Server Administrator screen. The help system displays a description of the parameters for the screen and a list of procedures you can perform from the screen. From this window, click **Show Navigation Area** to view the help system's table of contents from which you can search for a specific procedure or screen description.

The Server Configuration File

Configuration settings for the Integration Server are stored in the server configuration file (server.cnf). This file resides in the *Integration Server_directory* \instances*instance_name*\config directory and contains parameters that determine how the server operates.

Typically, you will use the Integration Server Administrator to set parameters in the server.cnf file, but there may be times when you need to edit the file directly with a text editor.

For a list of parameters in the server.cnf file and their default values, see [“Server Configuration Parameters” on page 1017](#).

Software AG Command Central

Software AG Command Central is a tool that release managers, infrastructure engineers, system administrators, and operators can use to perform administrative tasks from a single location. Command Central can assist with the following configuration, management, and monitoring tasks:

- Infrastructure engineers can see at a glance which products and fixes are installed, and where. Engineers can also easily compare installations to find discrepancies.

- System administrators can configure environments using a single web UI, command-line tool, or API so maintenance can be performed with a minimum of effort and risk.
- Release managers can prepare and deploy changes to multiple servers using command-line scripting for simpler, safer lifecycle management.
- Operators can monitor server status and health, as well as start and stop servers from a single location. They can also configure alerts to be sent to them in case of unplanned outages.

For more information about Software AG Command Central, see the Command Central documentation.

6 Managing Users and Groups

■ Users and Groups	84
■ Defining a User Account	84
■ Disabling and Enabling User Accounts	96
■ Defining Groups	98

Users and Groups

Use the Integration Server Administrator to define user and group information to the server. The definition for a user contains the user name, password, and group membership. The definition for a group contains the group name and a list of users in the group. The server stores and maintains the information.

Alternatively, you can set up the webMethods Integration Server to access the information from an external directory if your site uses one of the following external directories for user and group information:

- Central user management
- Lightweight Directory Access Protocol (LDAP)

This chapter describes only how the Integration Server works when user and group information is defined internally. For information about using an external directory with the Integration Server, see [“Configuring a Central User Directory or LDAP” on page 635](#).

Purpose of Users and Groups

Integration Server uses user and group information to authenticate clients and determine the server resources that a client is allowed to access.

If the server is using basic authentication (user names and passwords) to authenticate a client, it uses the user names and passwords defined in user accounts to validate the credentials a client supplies.

After a client is authenticated (whether through basic authentication or client certificates), the server uses the group membership to determine if a client is authorized for the requested action, such as, using the Integration Server Administrator or invoking a service.

Access to the server's resources is controlled at the group level. By setting up users and groups, you can control who can:

- **Configure and manage the server.** Only users that are members of the Administrators group (administrator privilege) can access the Integration Server Administrator.
- **Create, modify, and delete services that reside on the server.** Only users that are members of the Developers group (developer privileges) can connect to the server from Software AG Designer.
- **Access services and files that reside on the server.** Access to services and files is protected at the group level.

Defining a User Account

When you create a user account on Integration Server, you specify a user name, password, and group membership.

- **User name.** A user name is a unique name that identifies a client. You can specify a user name that represents an actual person (e.g., "JDSmith" for John D. Smith), or you can specify a user name to represent applications, job functions, or organizations. For example, you might set up generically named user names such as "MktgPurchAgent," "MktgTimeKeeper," and so forth, to represent job functions.
- **Password.** A password is an arbitrary string of characters that you associate with a user name. The server uses the password when authenticating a client who has submitted a valid user name. For more information about authentication, see [“Customizing Authentication Using JAAS” on page 527](#).

A password is meant to be a secret code shared only by the server, the server administrator, and the owner of the user account. Its purpose is to give the server added assurance that a request is coming from a legitimate user. Only administrators can assign a password to a user name and change a password for an existing account. For additional security, the server hashes passwords before storing them.

Note: Integration Server also provides password digest as an authentication option.

- **Group membership.** The group membership identifies the groups to which a user belongs. Access to the server's resources is controlled at the group level:

Only users that are members of the Administrators group can configure and manage the server using the Integration Server Administrator. For more information about controlling access to the Integration Server Administrator, see [“FIPS 140-2 Compliance” on page 1190](#).

Only users that are members of the Developers group can connect to the server from Software AG Designer to create, modify, and delete services. For information, see [“Adding a Developer User” on page 89](#).

The server protects access to services and files using Access Control Lists (ACLs). You set up ACLs that identify groups that are allowed or not allowed to access a resource. For more information about protecting services and files, see [“Controlling Access to Resources with ACLs” on page 501](#).

Predefined User Accounts

The server has the following predefined user accounts:

User	Groups	Description
Administrator	Everybody Administrators Replicators	A user account that has administrator privileges. You can use the Administrator user account to access the Integration Server Administrator to configure and manage the server. The predefined password for the Administrator account is "manage".
Default	Everybody Anonymous	The server uses the information defined for the Default user when the client does not supply a userid.

User	Groups	Description
Developer	Everybody Developers	A user that can connect to the server from Software AG Designer to create, modify, and delete services that reside on the server. The predefined password for the Developer account is "isdev".
Replicator	Everybody Replicators	The user account that the server uses during package replication. For more information about package replication, see “Copying Packages from One Server to Another” on page 673. The predefined password for the Replicator account is "iscopy."

Important: Software AG strongly recommends that you change the passwords for all the predefined user accounts as soon as you complete installation of Integration Server. Otherwise, your server will be vulnerable to anyone who knows the default passwords that webMethods installs on its servers. Software AG also recommends a best practice of creating an individual account for each administrator and developer instead of using the predefined accounts. Disable the predefined accounts after creating individual accounts.

Adding User Accounts

Use the following procedure to add a user account for a user.

➤ To add a user account to the server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Add and Remove Users**.
4. In the **Create Users** section of the screen, specify the following information:

Parameter	Specify
User Names	A unique user name made up of a combination of letters, numbers, or symbols. You can specify one user name per line. Press ENTER to separate the lines.

Important:

User names are case sensitive. When you create a user account, type it *exactly* as you want the client to enter it.

Parameter	Specify
	<p>Note: The string "SAMLart" is a reserved word in Integration Server. Do not create an Integration Server user name consisting of this word.</p>
Password	<p>A password made up of a combination of letters, numbers, or symbols.</p> <p>A password is required and cannot be a null value or an empty string.</p> <p>Important: Passwords are case sensitive. Type these values <i>exactly</i> as you want the client to enter it.</p> <p>Be sure to select passwords that are difficult to guess. For example, use a mixture of upper- and lowercase letters, numbers, and special characters. Do not use a name, phone number, social security number, license plate or other generally available information.</p> <p>Re-Enter Password The same password again to make sure you typed it correctly.</p>

5. Select **Allow Digest Authentication** to use password digest as an authentication option.
6. Click **Create Users**.

Deleting User Accounts

When a user account is no longer needed, you can delete the account. However, before you delete the user account make sure to update any locations in Integration Server where the account is used, such as execution users for triggers or users tasks, user mappings for client certificates, and outbound connection configurations such as remote server aliases. For a more complete list of locations that a user account might be used, see

.

Keep the following points in mind when deleting user accounts:

- When you delete a user, Integration Server automatically removes the user from the members lists of all the groups to which it was assigned.
- The following built-in user accounts cannot be deleted: Administrator, Default, Developer, and Replicator.
- Deleting a user account does not remove all of the user account data or other identifying data that appears in places such as log files. For information about removing user data from log files, see

➤ **To delete a user account from the server**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Add and Remove Users**.
4. In the **Remove Users** section of the screen, select the user names for the user accounts you want to delete.
5. Click **Remove Users**. The server issues a prompt to verify that you want to delete the user account. Click **OK** to remove the user account.

Adding an Administrator User

A user has administrator privileges if he or she belongs to the Administrators group or to any other group added to the Allow List of the Administrators ACL. To determine if a user has administrator privileges, the server authenticates the user to obtain his or her user name. (For information about how the server determines the user name, see [“Customizing Authentication Using JAAS” on page 527](#).) After determining the user name, the server determines if the user belongs to a group that is allowed and does not belong to any group that is denied access by the Administrators ACL. If so, the server allows access to the Integration Server Administrator.

To grant administrator privileges to a user, you must assign that user to the Administrators group or to a group you have added to the Allow list of the Administrators ACL. In addition, you must make sure the user is not a member of a group that is denied access by the Administrators ACL.

Important:

The user to whom you want to grant administrative privileges must already have a user account on the Integration Server. If the user does not already have a user account, create one before you perform the following steps.


➤ **To grant administrative privileges to a user**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.

Under **Local User Management**, the **Groups** area of the screen (on the right) contains two lists. **Users in this Group** is a list of users currently in the selected group. **Remaining Users** is a list of users *not* currently in the selected group.

3. In the **Groups** area of the screen, in the **Select group** list, select **Administrators**.
4. In the **Remaining Users** list, select (highlight) the user or users to whom you want to grant administrator privileges.

To select additional users without deselecting currently selected users, press the CTRL key while you click on the users you want to select. To deselect a user, press the CTRL key while you click the currently selected entry.

5. After you have selected all the users you want to add to the group, click . The server moves the selected users to the **Users in this Group** list.
6. Click **Save Changes**.

Adding a Developer User

A developer can use Software AG Designer to view, create, modify, and delete packages, folders, services, and other elements that reside on the server. Before the server allows a connection from Designer, it ensures that the user has developer privileges.

A user has developer privileges if he or she belongs to the Developers group or to any other group added to the Allow List of the Developers ACL. To determine if a user has developer privileges, the server authenticates the user to obtain their user name. (For information about how the server determines the user name, see [“Customizing Authentication Using JAAS” on page 527](#).) After determining the user name, the server determines if the user belongs to a group that is allowed and does not belong to any group that is denied access by the Developers ACL. If so, the server allows the connection between Designer and the server to be established.

To grant developer privileges to a user, you must assign that user to the Developers group or to a group you have added to the Allow list of the Developers ACL. In addition, you must make sure the user is not a member of a group that is denied access by the Developer ACL.

Important:

List, Read, and Write ACLs are a mechanism for protecting against accidental tampering or destruction of elements. A developer making a deliberate attempt can bypass this mechanism. Do not rely on ACLs for protection in a hostile environment.

Important:

The user to whom you want to grant developer privileges must already have a user account on the Integration Server. If the user does not already have a user account, create one for the user before you perform the following steps.

➤ **To grant developer privileges to a user**


1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.

Under **Local User Management**, in the **Groups** area of the screen (on the right) contains two lists. **Users in this Group** is a list of users currently in the selected group. **Remaining Users** is a list of users *not* currently in the selected group.

3. In the **Groups** area of the screen, in the **Select group** list, select **Developers**.

4. In the **Remaining Users** list, select (highlight) the user or users to whom you want to grant developer privileges.

To select additional users without deselecting currently selected users, press the CTRL key while you click on the users you want to select. To deselect a user, press the CTRL key while you click the currently selected entry.

5. After you have selected all the users you want to add to the group, click . The server moves the selected users to the **Users Currently in this Group** list.
6. Click **Save Changes**.

Changing Passwords

You can change the password for your user account or other user accounts. Keep the following points in mind when changing passwords:

- Do *not* change a password if you are outside of the corporate firewall and you did not use SSL to connect to the Integration Server.
- Passwords cannot be null values or empty strings.
- Be sure to select passwords that are difficult to guess. For example, use a mixture of upper- and lowercase letters, numbers, and special characters. Do not use a name, phone number, social security number, license plate or other generally available information; the security of your system depends on it.
- You cannot use the Integration Server Administrator or Software AG Designer to administer users or groups stored in an external directory. This restriction includes changing the passwords of these users.

> To change a user's password

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. In the **Users** section of the screen, select the user name for the user whose password you want to change and click **change password**.
4. Enter the following information:

Parameter	Specify
New Password	The new password, made up of any combination of letters, numbers, or symbols. Null values and empty strings are invalid.

Important:

Parameter	Specify
	<p>Passwords are case sensitive. Type this value <i>exactly</i> as you want the client to enter it.</p> <p>Be sure to select passwords that are difficult to guess. For example, use a mixture of upper- and lowercase letters, numbers, and special characters. Do not use a name, phone number, social security number, license plate or other generally available information.</p>
Confirm Password	The same password again to make sure you typed it correctly.

5. Select **Allow Digest Authentication** to use password digest as an authentication option.
6. Click **Save Password**.

Configuring Password Requirements and Expiration Settings

For security purposes, you can specify password requirements for user accounts. Using Integration Server Administrator, you can specify the following

- Password length and formation requirements. For more information, see [“Setting Password Requirements” on page 91](#)
- Password expiration settings. For more information, see [“Setting Password Expiration Requirements” on page 93](#)
- Account locking settings. For more information, see [“Configuring Account Locking Settings” on page 95](#)

Setting Password Requirements

For security purposes, Integration Server places length and character restrictions on passwords for administrator and non-administrator users. Integration Server contains a default set of password requirements; however, you can change these with the Integration Server Administrator.

➤ To set password length and character requirements for non-Administrator users

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Password Restrictions**.
4. Click **Edit Password Restrictions**.
5. Fill in information in the following fields:

Field	Description	Default
Enable Password Change	Whether users are allowed to change their passwords. These users must have developer privileges.	Yes
Password Enforcement Mode	Whether Administrator users are allowed to choose passwords that are not impacted by the password restriction settings. For Administrator users, when this property is set to <code>Strict</code> , Integration Server enforces the password restrictions. When set to <code>Lax</code> , the password restrictions are not enforced. For non-administrators the restrictions are always enforced.	<code>Lax</code>
Minimum Password Length	Minimum number of characters (alphabetic characters, digits, and special characters combined) the password must contain.	8
Maximum Password Length	Maximum number of characters (alphabetic characters, digits, and special characters combined) the password must contain. Maximum number of characters that a password can have is 128.	64
Minimum Number of Uppercase Characters	Minimum number of uppercase alphabetic characters the password must contain.	0
Minimum Number of Lowercase Characters	Minimum number of lowercase alphabetic characters the password must contain.	0
Minimum Number of Digits	Minimum number of digits the password must contain.	0
Minimum Number of Special Characters (neither alphabetic nor digits)	Minimum number of special characters, such as asterisk (*), period (.), and question mark (?) the password must contain.	0
	Note: A password cannot begin with an asterisk (*).	
Maximum Number of Identical Characters in a Row	Maximum number of identical characters in a row a password can contain.	3
Number of Old Passwords to Remember (per user)	Maximum number of previously set passwords that Integration Server saves for a user (excluding the current password). You cannot choose a password that matches any of the stored passwords. Maximum number of saved passwords is 12.	0
	Note:	

Field	Description	Default
	<p>Setting a new value to this field takes effect immediately upon saving. All the previously saved passwords, corresponding to the old value set for this field, are deleted and cannot be retrieved.</p> <p>For example, If initially administrator sets the value of this field to five, then Integration Server saves the last five passwords. If the administrator changes the value of this field to three, then Integration Server maintains only the latest three password entries and removes the oldest two entries. Similarly, if administrator changes it to six, then Integration Server maintains six latest entries.</p>	





6. Click **Save Password Settings**.

Setting Password Expiration Requirements

For security purposes, Integration Server allows administrators to set password expiration requirements on passwords for administrator and non-administrator users. Integration Server contains a default set of password expiry setting; however, you can change these with the Integration Server Administrator. An administrator user receives a reminder email to reset the password before certain number of days, as specified in the Password Expiration Settings page of Integration Server Administrator.

> To configure password expiration settings

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Password Expiration Settings**.
4. Click **Edit Password Expiration Settings**.
5. To grant administrative privileges to users, perform the following:

Field	Description
Applies to Users	List of users to whom the expiration interval setting is applicable. Use the  and  buttons to move user names on or off this list.
Remaining Users	List of users to whom the expiration interval setting is not applicable. Use the  and  buttons to move user names on or off this list.

6. Fill in information in the following fields:

Field	Description	Default
Enabled	Whether to enable the password expiry settings. This option is disabled by default. Select Enabled to enable the password expiry settings.	
Expiration Interval	Number of days after which a password will expire, 90 if not changed. The value should be a non-zero integer. Note: Upon save, when this option is enabled, any password that a user had set before the expiration interval are considered expired and the user must reset them. For example, if you changed your password 10 days ago and now, the Administrator changes the Expiration Interval to 5 days, then your password has expired and needs to be reset.	
Expiration Notice Email Addresses	List of email addresses to which Integration Server sends an email notification informing that the user password is about to expire or has already expired. If the field is left blank, then Integration Server uses the email address specified in the Internal Email field of Email Notification section on Resource Settings Screen (Settings > Resources). Note: Integration Server uses the SMTP server and port details specified in the Email Notification section on Resource Settings Screen (Settings > Resources).	
Expiration Email Reminders	Specifies the number of days prior to password expiry that Integration Server should start sending	3

Field	Description	Default
	<p>the reminder emails for password reset. The emails are sent daily until the user either updates the password or changes the expiration interval. Set the value to 0 to prevent Integration Server from sending the reminder emails for soon to expire passwords. Default is 3.</p> <p>Note:Integration Server continues to send reminder emails until the user resets the password, even after the password has expired. Reminder emails are sent for expired passwords, even if you set the value to 0.</p>	

7. Click **Save Changes**.

Configuring Account Locking Settings

For security purposes, it is important to lock a user account when the user fails to provide the correct password after a specified number of failed login attempts to Integration Server. A locked user account remains locked for a specific period of time, after which the account gets unlocked. Integration Server allows administrators to configure the account locking settings for administrator and non-administrator users. Using Integration Server Administrator, you can set the values for number of attempts by a user before locking the account and also the duration of the lock interval. Integration Server also allows administrators to manually unlock locked user accounts.

➤ To configure account locking settings

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Account Locking Settings**.
4. Click **Edit Account Locking Settings**.
5. Fill in information in the following fields:

Field	Description	Default
Enabled	Whether to enable the account locking settings. Select Enabled to enable the account locking settings.	


Field	Description	Default
Maximum Login Attempts	Number of attempts in the specified time interval (minutes, hours, or days) to provide the correct password before locking the account.	None
Lockout Duration	Duration for which the account remains locked.	None
Apply Account Locking Policy To	List of users to whom the account locking settings apply. Specify one of the following: <ul style="list-style-type: none">■ All Users except Predefined Users- indicates that account locking rules apply to all user accounts except the predefined user accounts (Administrator, Default, Developer, and Replicator).■ All Users - indicates the account locking rules apply to all user accounts.	All Users except Predefined Users

6. Click **Save Changes**.

Unlocking User Accounts

Integration Server unlocks a user account after the specified locked duration. However, you can use Integration Server Administrator to manually unlock user accounts.

> To unlock user accounts

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Account Locking Settings**.
4. Click **Locked Users**.
5. From the list of locked user accounts, click  to unlock the user account.
6. Click **Save Changes**.

Disabling and Enabling User Accounts

There may be times when you need to disable a user account. Doing so makes password cracking attacks harder by eliminating well-known user names. When you disable a user account, login

attempts with that user name will fail authentication and be rejected. For example, you might disable the user account of a developer who is on vacation, or the account of a trading partner whose trading privileges are suspended. Because the user has been disabled rather than deleted, you can later reinstate the account without changing the password or resetting permissions.

For deployment, you should disable the Administrator user to prevent someone from trying to guess the password and gain access to your system. Before disabling the Administrator user, you must first create another user, for example SmithAdmin, and add it to the Administrators, Developers, and Replicators groups. Then disable the Administrator user. (Internal server functions that run as the Administrator user, such as start up and shut down services, will still be able to run as Administrator.) Then you can use the SmithAdmin user to administer your Integration Server.

Disabling a User Account

Use the following procedure to disable a user account.

Important:

Before you disable the Administrator user, make sure you have defined another user with administrator privileges so you are not locked out of the server.

> To disable a user account

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Enable and Disable Users**.
4. In the **Enabled Users** list select (highlight) the user or users you want to disable.

To select additional users without deselecting currently selected users, press the CTRL key while you click on the users you want to select. To deselect a user, press the CTRL key while you click the currently selected entry.

5. At the bottom of the **Enabled Users** area of the screen click .

The server moves the selected users to the **Disabled Users** area of the screen.

6. Click **Save Changes**.

Enabling a User Account

Use the following procedure to enable a user account. The only time you will need to enable a user account is if the system administrator explicitly disabled it.

> To enable a user account

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Enable and Disable Users**.
4. In the **Disabled Users** list select (highlight) the user or users you want to enable.

To select additional users without deselecting currently selected users, press the CTRL key while you click on the users you want to select. To deselect a user, press the CTRL key while you click the currently selected entry.

5. At the bottom of the **Disabled Users** area of the screen click .

The server moves the selected users to the **Enabled Users** area of the screen.

6. Click **Save Changes**.

Defining Groups

A group is a named collection of users that share privileges. The privileges can be:

- Administrator privileges
- Replicator privileges
- Developer privileges
- Privileges to invoke a service
- Privileges to allow the server to serve files

Privileges to invoke a service or access files are granted and denied by Access Control Lists (ACLs) that you set up. When an administrator creates ACLs, he or she identifies groups that are allowed to access services and files and groups that are denied access to services and files.

Administrator, replicator, and developer privileges are typically granted by adding a user to the Administrators, Replicators, or Developers group, respectively. Alternatively, you can create new groups and add them to the allow lists of the Administrators, Replicators, or Developers ACLs.

Create groups that identify groups of users that will share the same privileges. When you create a group definition, you specify a group name and the members of the group.

- **Group name.** A group name is a unique name that identifies the group. You can use any name, for example, a name that defines a department (Marketing) or job function (Programmers).
- **Members.** List of user names that are members of the group.

Predefined Groups

Integration Server is installed with the following predefined groups.

Group Name	Members	Description
Administrators	Administrator	<p>This group identifies users that have administrator privileges. A user must have administrator privileges to configure and manage the server.</p> <p>Important: Membership in this group gives substantial power to affect the configuration of the Integration Server. Use caution in assigning membership in this group to individuals who can be trusted to use the privilege carefully.</p>
Anonymous	Default	This group identifies users that have not supplied a userid.
Developers	Developer	<p>This group identifies users that have developer privileges. A user must have developer privileges to connect to the server from the Designer.</p> <p>Important: Membership in this group gives substantial power to affect the configuration of the Integration Server. Use caution in assigning membership in this group to individuals who can be trusted to use the privilege carefully.</p>
Everybody	Administrator Default Developer Replicator	All users are a member of this group. Every new user is automatically added to the Everybody group.
Replicators	Administrator Replicator	<p>This group identifies users that have replicator privileges. The Replicators group gives its members the authority to perform package replication. (By default, the server uses members of the Replicators group for package replication.)</p> <p>Users do not have to be members of the Replicators group to perform package replication. As long as user is a member of a group that is assigned to the Replicators ACL, it can perform package replication.</p> <p>For more information about package replication, see “Copying Packages from One Server to Another” on page 673.</p> <p>Membership in this group gives substantial power to affect the configuration of the Integration Server. Use caution in assigning membership in this group to individuals who can be trusted to use the privilege carefully.</p>

Adding Groups

Use the following procedure to add groups.

> To add a new group to the server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Add and Remove Groups**.
4. In the **Create Groups** area of the screen, type a unique group name made up of a combination of letters, numbers, or symbols.

Important:

Group names cannot contain spaces and special characters such as comma (,), quotation marks (' or "), backslash (\), and forward slash (/).

You can add more than one group at a time by specifying multiple lines, one group to a line. Press ENTER to separate lines.

5. Click **Create Groups**.

Adding Users to a Group

Use the following procedure to add users to a group.

Note:

You cannot change the membership of the Everybody group.

> To add users to a group

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.


The server displays the following screen.

The screenshot shows the 'Users' and 'Groups' management interface. The 'Users' section on the left has a 'Select User:' dropdown set to 'Administrator' with a '(change password)' link below it. Below this are two lists: 'Groups user belongs to' containing 'Administrators', 'Everybody', and 'Replicators'; and 'Remaining Groups' containing 'Anonymous' and 'Developers'. The 'Groups' section on the right has a 'Select group:' dropdown set to 'Administrators'. Below this are two lists: 'Users in this Group' containing 'Administrator'; and 'Remaining Users' containing '-----none-----', 'Default', 'Developer', and 'Replicator'. Navigation arrows are present between the lists, and a 'Save Changes' button is at the bottom left.

The **Groups** area of the screen (on the right) contains two lists. **Users in this Group** is a list of users currently in the group. **Remaining Users** is a list of users *not* currently in the group.

3. Under **Groups**, in the **Select group** list, select the group to which you want to add a user.
4. In the **Remaining Users** list select (highlight) the user or users you want to add to the group.

To select additional users without deselecting currently selected users, press the CTRL key while you click on the users you want to select. To deselect a user, press the CTRL key while you click the currently selected entry.

5. After you have selected all the users you want to add to the group, click . The server moves the selected users to the **Users Currently in this Group** list.
6. Click **Save Changes**.

Removing Users from a Group

Use the following procedure to remove users from a group.

Note:

You cannot change the membership of the Everybody group.

> To remove a user from a group

1. Open the Integration Server Administrator if it is not already open.

- In the **Security** menu of the Navigation panel, click **User Management**.

The server displays the following screen.

The screenshot shows a web interface for managing users and groups. It is divided into two main panels: **Users** (left) and **Groups** (right).

Users Panel:

- Select User:** A dropdown menu showing "Administrator" with a "(change password)" link below it.
- Groups user belongs to:** A list box containing "Administrators", "Everybody", and "Replicators".
- Remaining Groups:** A list box containing "Anonymous" and "Developers".
- Two blue buttons with right and left arrows are positioned below the lists.

Groups Panel:


- Select group:** A dropdown menu showing "Administrators".
- Users in this Group:** A list box containing "Administrator".
- Remaining Users:** A list box containing "-----none-----", "Default", "Developer", and "Replicator".
- Two blue buttons with right and left arrows are positioned below the lists.

At the bottom left of the interface is a "Save Changes" button.

The **Groups** area of the screen (on the right) contains two lists. **Users in this Group** is a list of users currently in the group. **Remaining Users** is a list of users *not* currently in the group.

- Under **Groups**, in the **Select group** list, select the group from which you want to remove a user.
- In the **Users in this Group** area of the screen, select (highlight) users that you want to remove from the group.

To select additional users without deselecting currently selected users, press the CTRL key while you click on the users you want to select. To deselect a user, press the CTRL key while you click the currently selected entry.

- At the bottom of the **Users in this Group** area of the screen click . The server moves the selected users to the **Remaining Users** area of the screen.

Viewing Group Membership

Use the following procedure to view the members of a group or change the members in a group.

- > **To view group membership for a group**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel click **User Management**.

The server displays the following screen.

The **Groups** area of the screen (on the right) contains two lists. **Users Currently in this Group** is a list of users currently in the selected group. **Remaining Users** is a list of users *not* currently in the selected group.

3. Under **Groups**, in the **Select group** list, select the group for which you want to view membership.
4. The server displays the users in the **Users in this Group** list.

Removing Groups

Use the following procedure to remove groups that you no longer need.

Note:

You cannot delete any of the following groups: Administrators, Developers, Replicators, Anonymous, and Everybody.

> To delete a group from the server

1. Open the Integration Server Administrator if it is not already open.

2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Add and Remove Groups**.
4. In the **Remove Groups** area of the screen, select the groups you want to remove.
5. Click **Remove Groups**.

7 Configuring the Server

■ Viewing and Changing Licensing Information	106
■ Managing the Server Thread Pool	109
■ Managing Server Sessions	110
■ Configuring Outbound HTTP Settings	112
■ Setting Up Aliases for Remote Integration Servers	115
■ Specifying Third-Party Proxy Servers for Outbound Requests	118
■ Configuring Where the Integration Server Writes Logging, Status, and Other Information	127
■ Switching from the Embedded Database to an External RDBMS	127
■ Working with Extended Configuration Settings	127
■ Specifying Character Encoding	129
■ Configuring the JVM	129
■ Specifying the JDK or JRE for Integration Server	129
■ Changing the JVM Heap Size Used by Integration Server	130
■ Publishing and Retracting Information about Integration Server Assets	131
■ Setting a Port for Remote Client JMX Monitoring	133
■ Configuring Integration Server to Accept a Debug Connection During Startup	134
■ Using CORS with Integration Server	135

Viewing and Changing Licensing Information

When you purchase a webMethods Integration Server, your organization is granted a license to use it with certain features and functionality and with a specified number of concurrent users (simultaneous sessions).

The license expires after a time period specified by your particular purchase agreement.

The License Key

Before you install Integration Server, you are provided with a license key file that you place in the file system of the machine on which Integration Server will run. This file contains your license *key*, which is a special code associated with your license.

When you install Integration Server, the setup program asks you to provide the name and location of this file. The setup program then copies this file to the *Integration Server_directory* \instances*instance_name*\config directory with the name licenseKey.xml. If this file is inadvertently deleted, Integration Server reverts to demo mode. In this mode, there are only two licensed sessions and the server automatically shuts down 30 minutes after it is started.

Viewing Licensing Information

To view licensing information or change the license key for your Integration Server, use the **Licensing** screen in the Integration Server Administrator.

> To view licensing information

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Licensing**.

Integration Server displays a list of features, with a check mark next to each feature you are licensed to use.

3. To view more detailed licensing information, click **Licensing Details**.

Integration Server displays detailed information, including the actual license key, the list of features you are licensed to use, the maximum number of concurrent sessions allowed to run on the server, and partner and Trading Networks information.

Changing Licensing Information

When your license expires or you change your license to include different features, you can use Integration Server Administrator to change the license file for Integration Server.

For information about changing the license file used with an Integration Server image running in a Docker container, see the description of the SAG_IS_LICENSE_FILE ENV variable in [“Environment Variables for Use with Docker” on page 1183](#).

Note:

Before changing the licensing information in Integration Server Administrator you must obtain a new license key file from Software AG and copy it to the file system of the machine on which Integration Server runs.

Use the following procedure to change the license key used with Integration Server.

> To change the License Key

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Licensing**.
3. Click **Licensing Details**.
4. Click **Edit Licensing Detail**.

The server displays the **Edit** screen.

5. In the **Integration Server License Key File** field, enter the pathname for the license key file you obtained from Software AG.
6. Click **Save Changes**.

Integration Server copies the contents of the license key file to *Integration Server_directory \instances \instance_name \config \licenseKey.xml* and updates the **Integration Server License Key File** field to reflect that name.

Note: Integration Server updates the expiration date automatically after you click **Save Changes**. It is not necessary to restart Integration Server.

Renewal Reminders

Approximately 30 days before your license expires, Integration Server sends an e-mail message to the administrative-message recipient, reminding him or her to renew the license. In addition, the server displays the following message at the top of all pages on the Integration Server Administrator:

```
License key expires in about xxx days ... contact Software AG for a new key.
```

Renewing a Key

If you need to obtain a new key or renew your license, contact your Software AG sales representative.

Adding Licensed Features

Available features are based on your contract with Software AG. To add features to your contract, contact your Software AG sales representative.

Licensed Sessions

Your license allows a specified number of users to have sessions in the Integration Server concurrently. The Integration Server creates a session when a developer connects to the server from Software AG Designer or an IS client connects to the server to execute services. If a user attempts to access the server while the maximum number of sessions are in use, the server rejects the request and returns the following error to the user:

```
Server has reached client limit.
```

You can view the current number of active sessions and the licensed session limit using the **Statistics** screen in the Integration Server Administrator. This value is permanently associated with your license key and can only be changed by obtaining a new license.

Any connection made to the server by a non-Administrator user (that is, a user that is not part of the Administrators group) consumes a licensed session. The session exists until it times out (based on the server's Session Timeout setting) or the requester stops the session by invoking the `wm.server.disconnect` service.

If a user invokes a stateless service and a session does not already exist for the user, the server creates a session. If the user is a non-Administrator, the user consumes a licensed session. After the service completes, the server removes the session and reduces the number of licensed sessions in use.

Note:

If Integration Server receives multiple requests simultaneously and does not have the resources to handle them, the server's performance might decrease. You can tune performance by adjusting the concurrent stateful sessions limit on the server. For more information, see [“Managing Server Sessions” on page 110](#).

Viewing Active Sessions

Use the following procedure to view the number of active sessions and to check the licensed sessions limit.

➤ To view the number of active sessions and the licensed sessions limit

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Statistics**.

The server displays the current number of active sessions in use in the **Total Sessions** field. The server displays the maximum number of licensed sessions your license allows in the **Licensed Sessions** fields.

For detailed information about the active sessions, click the number in the **Total Sessions** field.

Managing the Server Thread Pool

To better tune your server's performance, you can configure the minimum and maximum number of threads. The server uses threads to execute services, retrieve documents from the messaging provider, and execute triggers. When the server starts, the thread pool initially contains the minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed. If this maximum number is reached, the server waits until processes complete and return threads to the pool before beginning more processes.

You can also set a warning level for available threads. When the percentage of available threads is equal to or less than the warning level, the server generates a journal log message to alert you to the reduced thread availability. The server generates another journal log message when the number of available threads is greater than the threshold.

To view system threads that are running on the server, navigate to the **Server > Statistics > System Threads** screen. See [“Canceling or Killing a Thread” on page 709](#) for more information.

➤ To configure the server thread pool

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Resources**.
3. Click **Edit Resource Settings**.
4. Under **Server Thread Pool**, update the server thread pool settings, as follows:

For this parameter...	Specify...
Maximum Threads	The maximum number of threads that the server maintains in the server thread pool. If this maximum number is reached, the server waits until processes complete and return threads to the pool before running more processes. The default is 75.
Minimum Threads	The minimum number of threads the server maintains in the server thread pool. When the server starts, the thread pool initially contains this minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed, which is specified in the Maximum Threads field. The default is 10.
Available Threads Warning Threshold	Threshold at which the server starts to warn of insufficient available threads. When the percentage of available server threads equals this percentage, the server generates a Journal log message indicating the current available thread percentage and stating:

For this parameter...**Specify...**

Available Thread Warning Threshold Exceeded

The default is 15%.

When you enter a percentage and save your changes, the server automatically calculates the number of threads and displays the number next to the specified percentage.

Tip:

When the percentage of available threads falls below the warning level, you might want to decrease the number of documents the server receives and processes for webMethods messaging triggers. For more information, see [“Managing webMethods Messaging Triggers” on page 827](#).

Scheduler Thread Throttle Percentage of server threads the scheduler function is permitted to use. The default is 75%.

5. Click **Save Changes**.

Managing Server Sessions

Integration Server starts a new session for every remote client that connects to it. If a session becomes idle or inactive for a long period of time or if too many stateful sessions are created simultaneously, the server's performance might be affected.

Integration Server provides various controls that you can use to manage sessions. Specifically, you can:

- Limit the number of minutes an idle session can remain active.
- Limit the number of stateful sessions that can be created on the server concurrently.
- Set a warning level for available stateful sessions.

Setting the Session Timeout Limit

Once a session starts, it remains active until the client application specifically issues a disconnect instruction to the server (which forces an immediate termination) or the session "times out" due to inactivity, whichever comes first.

If a session is idle for a long period of time, it usually means that the client is no longer active or that the connection between client and the server has been lost. The server constantly monitors for inactive sessions, and terminates sessions that are idle for more than the allowed period of time. If the server did not take steps to clear out such sessions, they would remain active indefinitely, wasting valuable server resources.

➤ To set the session timeout limit

1. Open the Integration Server Administrator if it is not already open.

In the **Settings** menu of the Navigation panel, click **Resources**.

2. Click **Edit Resource Settings**.
3. Under **Session** in the **Session Timeout** field, enter maximum number of minutes an idle session can remain active (in other words, how long you want the server to wait before terminating an idle session).

To set the **Session Timeout** parameter appropriately, you must be familiar with the clients that use your server. If your clients are all Java programs, you can usually reduce the timeout value to 6 or 7 minutes. You may need to experiment with this setting to find the appropriate value for your site. By default, the server uses a timeout limit of 10 minutes. This is an appropriate value for most sites. However, you may have to increase this value if your clients normally have lengthy delays (greater than 10 minutes) between successive requests.

4. Click **Save Changes**.

Setting the Stateful Session Limit

Integration Server starts a new session for every remote client that connects to it. This can be a problem if the server receives multiple requests simultaneously and does not have the resources to handle them.

The number of concurrent sessions allowed is specified by your license. However, you can tune performance by setting the stateful session limit using the **Resources** screen in Integration Server Administrator. When you set a stateful session limit and the number of concurrent stateful sessions exceeds that limit, the server rejects new requests and returns an error message to the user.

You can also set a warning level for available stateful sessions. When the percentage of available stateful sessions is equal to or less than the warning level, the server generates a message in the server log to alert you of stateful session use and availability.

➤ To set the stateful sessions limit

1. Open the Integration Server Administrator if it is not already open.

In the **Settings** menu of the Navigation panel, click **Resources**.

2. Click **Edit Resource Settings**.
3. Under **Session**, update the server session settings, as follows:

For this parameter...	Specify...
Enable Stateful Session Limit	<p>Whether you want Integration Server to limit the number of concurrent stateful sessions. If you want to enable a stateful session limit, select Yes; otherwise, select No.</p> <p>When enabled, the number of stateful sessions that can exist simultaneously on the server is defined by the Maximum Stateful Sessions parameter.</p> <p>You can view statistics for stateful sessions on the Server > Statistics screen in Integration Server Administrator. When disabled, statistics for stateful sessions are still gathered and displayed on the Server > Statistics screen.</p>
Maximum Stateful Sessions	<p>The maximum number of concurrent stateful sessions that can exist on the Integration Server. If a user attempts to access the server and execute a stateful service while the maximum number of stateful sessions are in use, the server rejects the request and returns the following error to the user:</p> <pre>Server is not accepting new requests at this time.</pre> <p>The value must be a positive integer. If a value is not specified or the feature is disabled, the maximum number of concurrent sessions is determined by the licensed sessions limit specified in your Integration Server license file.</p>
Available Stateful Sessions Warning Threshold	<p>Threshold at which Integration Server starts to warn of insufficient available stateful sessions. When the percentage of available stateful sessions equals or falls below the value of this property, Integration Server generates a server log message stating:</p> <pre>{0}% or more of maximum number of concurrent stateful sessions are in use. {1} sessions are available</pre> <p>The default is 25%.</p>

4. Click **Save Changes**.

Configuring Outbound HTTP Settings

Outbound HTTP parameters control how the server presents and processes outbound HTTP and HTTPS requests (i.e., requests that Integration Server issues on behalf of a client). The parameters control behavior such as how long the server waits for a response, how many times it retries a failed request, and so forth.

Developers can override some of the server's outbound HTTP setting defaults at run time, as described below.

Parameter	Description
User Agent	<p>Specifies the value that the server uses in the HTTP User Agent request header that it sends when requesting a web document. The User Agent header tells a web server what type of browser is making the request. In the case of the Integration Server, the User Agent header indicates the type of browser that the Integration Server appears to be to the web server. Some web servers examine this header to determine a client's capabilities so they can tailor their responses accordingly.</p> <p>When you install the Integration Server, the User Agent parameter is set to Mozilla/4.0 [en] (WinNT; I). You can change this value as you need; however, the value you set should satisfy the majority of services that your server executes.</p> <p>Be sure your developers know the User Agent value your server uses. If their applications require a different User Agent, they can override the server's default at run time by including an HTTP User Agent header with their request.</p>
Maximum Redirects	<p>Specifies the number of times that the Integration Server allows a request to be redirected (i.e., automatically sent to another URL by the target server. If a request exceeds the specified number of redirections, the Integration Server immediately returns an I/O exception to the client.</p> <p>When you install the Integration Server, Maximum Redirects is set to 5. You will need to increase this value if the targets that you access typically redirect their requests more than this. (This may happen if the target operates in a clustered environment.)</p>
Timeout	<p>Specifies the length of time the server waits for a response from a target server. If the Integration Server does not receive a response in the allotted time, it retries the request up to the number of times specified by the Retries parameter. When the allowed number of retries is exceeded, the server returns an exception.</p> <p>When you install the Integration Server, the Timeout parameter is set to 300 seconds (5 minutes). To set Integration Server to wait indefinitely for a response from the target server, set this parameter to 0.</p> <div style="background-color: #f0f0f0; padding: 5px;"><p>Important: If you set the Timeout parameter to 0 and the target server does not respond to the request, the Integration Server making the request cannot process new requests due to thread pool exhaustion.</p></div> <p>You can also specify number of seconds the server waits for an HTTP request to be fulfilled using the <code>watt.net.timeout</code> server configuration parameter. For more information, see “Server Configuration Parameters” on page 1017.</p>

Parameter	Description
Retries	<p>Specifies the number of times the server reissues a request that has timed out (i.e., one from which it did not receive a response within the time period specified by the Timeout parameter).</p> <p>When you install the Integration Server, Retries is set to 0. This means that the server automatically returns an exception if it does not get a response within the allotted time. Set Retries to a value greater than 0 if you want the server to retry (reissue) timed-out requests. The server will retry the request the number of times you specify.</p> <p>Make sure that your developers know the Retries value that your server uses. If they need to use a different value, they can explicitly assign a Retries value to their service.</p>

Specifying Outbound HTTP Settings

Use the following procedure to specify the Outbound HTTP Settings.

➤ To set the Outbound HTTP Settings

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Resources**.
3. Click **Edit Resource Settings**.
4. Set the **Outbound HTTP Settings** as follows:

For this parameter...	Specify...
User Agent	The string that you want the server to supply in the HTTP User Agent header if the client does not specify a value. Type the string exactly as you want it to appear in the HTTP header, including spaces, symbols, and punctuation.
Maximum Redirects	An integer that indicates the number of times to allow a request to be redirected before the server returns an I/O exception to the client.
Timeout	An integer that indicates the number of seconds the server waits for a response from the target server before retrying the service or returning a timeout error to the client.
Retries	An integer that indicates the number of times the server retries a service that has timed out before returning an exception to the client.

5. Click **Save Changes**.

Setting Up Aliases for Remote Integration Servers

You can set up aliases for remote servers. Communication through the alias is optimized, making transactions with the remote server faster.

Use a remote alias when:

- **Invoking services on other Integration Servers**. After you establish aliases, you can use the `pub.remote:invokeand` and `pub.remote.gd:*services` to invoke services on remote servers by identifying the remote servers by their aliases.
- **Presenting multiple client certificates**. The Integration Server can present a single client certificate to all servers or it can present different client certificates to different SSL servers. In addition, the Integration Server can present certificates provided for this purpose by other organizations. Setting up remote aliases for these SSL servers makes it easier to present different certificates to them. See [“Using Multiple Client Certificates” on page 523](#) for more information.
- **Performing package replication**. For a subscriber to set up a subscription with a publisher or pull a package from the publisher, you must define the publishing server as a remote server to the subscriber. The alias tells the subscribing server how to connect to the publishing server to set up the subscription or pull the package. See [“The Subscribing Server” on page 687](#) for more information.

The definition for an alias contains the connection information the server requires to connect to a remote server. It identifies the host name or IP address of the remote server and indicates whether the server should use an HTTP or HTTPS connection to connect to the remote server.

The alias also identifies a user name and password that the server supplies to the remote server. The remote server uses the user name and password to authenticate the client and to determine if the client is authorized to execute the requested service.

In effect, the alias grants access to a remote service by allowing the user to impersonate an authorized user on the remote server. Therefore, to prevent unauthorized users from accessing services on remote servers, the alias also contains access control information. You specify an ACL that protects the use of the alias. If a client that is authorized to use the alias makes a request, the server will request the service on the remote server. If a client that is not authorized to use the alias makes a request, the server rejects the request and does not invoke the service on the remote server.

Adding an Alias for a Remote Integration Server

Use the following procedure to add an alias for a remote Integration Server.

➤ To add an alias for a remote server

1. Open the Integration Server Administrator if it is not already open.

- In the **Settings** menu of the Navigation panel, click **Remote Servers**.
- Click **Create Remote Server Alias**.
- Set the **Remote Server Alias Properties** as follows:

For this parameter...	Specify...
Alias	Name that you want to use for the alias. You can give the remote server any alias name but it cannot include the following illegal characters: <code>#-&@^!%*:\$./\ \ ` ; , ~ + =) ({ } [] > < " ' .</code>
Host Name or IP Address	Host name or IP address of the remote server for which you are creating an alias (e.g., workstation5.webmethods.com). Note: This field does not accept white-space characters (or blank spaces).
Port Number	Port number on which the remote server listens for incoming requests from your server (e.g., 5555).
User Name	User name for a user account on the remote server. When you invoke a service using this alias, the remote server uses this user account for authentication and access control. Specify a user name that has access to the services you want to invoke on the remote server.
Password	Password identified in the user account for User Name .
Execute ACL	ACL that governs which user groups on your server can use this alias for the remote server. Select an ACL from the drop down list. By default, only members of groups governed by the Internal ACL can use this alias.
Max Keep Alive Connections	Sets the default number of client keep alive connections to retain for a given target endpoint. If not specified, five keep alive connections are retained. This field specifies the maximum number of client connections that should be retained for any given remote host. In other words, this is not a maximum number of connections that can be established, but a limit on the number of inactive client connections to retain for reuse.
Keep Alive Timeout	Specifies the length of time (in minutes) that your server maintains an idle connection to a remote server. This value will cause the connection to be retained for possible reuse until it times out. If the specified keep alive timeout value expires, the connection will close and the HTTP Listener will attempt to create a new one.
Use SSL	Whether you want your server to connect to the remote server using Secure Sockets Layer (SSL). If you want to use SSL, select yes ; otherwise, select no .

For this parameter...	Specify...
	<p>Important: If you select yes, the remote server must be configured to listen for incoming HTTPS requests.</p>
Keystore Alias (optional)	<p>A user-specified, text identifier for an Integration Server keystore.</p> <p>The alias points to a repository of keys and their associated SSL certificates.</p>
Key Alias (optional)	<p>The alias for the Integration Server private key and associated certificate, which must be stored in the keystore specified by the above keystore alias.</p> <p>You must create a key alias with a third-party certificate utility, such as Java keytool. You cannot create a key alias with Integration Server Administrator.</p>
Retry Server	<p>Host name or IP address (for example, workstation6.webmethods.com) of a remote server you want your local Integration Server to connect to if the primary remote server is unavailable. The retry server you specify will use the same port as the primary remote server. If the remote server is part of a cluster, the local Integration Server will, by default, try to connect to other Integration Servers in the cluster before trying to connect to the retry server. If clients are using the <code>pub.remote:invoke</code> service to run services on a remote server, it is possible to change this default behavior by using the <code>\$retryCluster</code> input parameter with the service. If you set this parameter to false, the service will not try to use other Integration Servers in the cluster. Instead, the service will immediately try using the retry server specified on this screen.</p>

5. Click **Save Changes**.

Testing the Connection to a Remote Server

After you add an alias, you can test the connection to the remote server to ensure that the host name (or IP address) and port number specified for the alias identifies an Integration Server that is currently running. Use the following procedure to test the connection.

> To test the connection to a remote server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Remote Servers**.
3. Click the ► icon in the **Test** column for the alias you want to test.

The server displays a status line that indicates whether the connection is successful or not. The status line is displayed above the list of existing aliases.

Editing an Alias

If you need to update the information for an alias, you can edit it to make your changes. Use the following procedure to edit an alias.

➤ To edit an alias for a remote server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Remote Servers**.
3. Locate the alias you want to edit and click on the alias name.
4. Update the information for the alias.
5. Click **Save Changes**.

Deleting an Alias

If you no longer need an alias for a remote server, you can delete it. Use the following procedure to delete an alias.

➤ To delete an alias for a remote server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Remote Servers**.
3. Locate the alias you want to delete and click the **✗** icon in the **Delete** field. The server displays a dialog box that prompts you to verify your action. Click **OK** to verify that you want to delete the alias.

Specifying Third-Party Proxy Servers for Outbound Requests

When Integration Server executes a request against a remote server, it issues an HTTP, HTTPS, FTP, or SOCKS request to the specified target server. For example, Integration Server might invoke a service on a remote Integration Server or execute a web service connector that invokes a web service. If your Integration Server sits behind a firewall, and must route these HTTP, HTTPS, FTP, or SOCKS requests through a third party proxy server, you can use Integration Server Administrator to identify one or more proxy servers to which Integration Server will route these requests.

For Integration Server to use a proxy server, you must define a *proxy server alias*. The proxy server alias identifies a proxy server and a port on the server through which you want to route requests.

You can configure Integration Server to route requests to one or more proxy server aliases for each type of outbound request (HTTP, HTTPS, FTP, and SOCKS).

You can specify a default proxy server for each protocol type. Integration Server uses the default proxy server alias when one is not specified by the `pub.client:http`, `pub.client:ftp.login`, or `pub.client:ftp` service. Integration Server also uses the default proxy server alias when executing a web service connector associated with an HTTP/S consumer web service endpoint alias that does not specify a proxy alias.

Note:

If you have configured and enabled a SOCKS proxy server alias, it acts as a fallback proxy server for HTTP, HTTPS, and FTP proxy server aliases. Even if a SOCKS proxy is set as the default proxy server alias, Integration Server gives precedence to the configured HTTP, HTTPS, and FTP proxy server aliases for the corresponding outbound requests before attempting to connect through the SOCKS proxy alias.

How Integration Server Uses Proxy Servers

When Integration Server sends a request to a remote server, Integration Server routes the request through a proxy server unless the domain of the target server is listed as a proxy bypass. When a domain appears on the proxy bypass list, Integration Server sends requests directly to the target server. If a domain is not on the proxy bypass list, the proxy server that Integration Server uses to send the request depends on the following conditions:

- Whether the request specifies a proxy server alias
- Whether a default proxy server alias exists and is enabled.

The following table describes how Integration Server determines which proxy server to use when sending a request. This process applies to the HTTP, HTTPS, FTP, and SOCKS protocols.

Proxy Server Alias Specified?	Default Proxy Server Alias Exists?	Action taken by Integration Server
Yes	NA	<p>If the specified proxy server alias is enabled, Integration Server sends the request using the proxy server in the specified proxy alias. If the send attempt fails, Integration Server does not attempt to make a direct connection to the remote server. Also, Integration Server does not use the proxy server specified in the default proxy alias or any other proxy server to send the request.</p> <p>If the specified proxy server alias is disabled, the send request fails.</p>

Proxy Server Alias Specified?	Default Proxy Server Alias Exists?	Action taken by Integration Server
No	Yes	<p>If the default proxy server alias is enabled, Integration Server sends the request using the proxy server in the default proxy server alias. If the send attempt fails, Integration Server either sends the request to the remote server using a direct connection or throws an exception depending on the settings specified for the <code>watt.net.proxy.fallbackToDirectConnection</code> parameter.</p> <p>If the default proxy server alias is not enabled, Integration Server sends the request using a proxy server for any of the configured proxy server aliases. If the attempt to send a request made using that proxy server fails, Integration Server tries to send using another proxy server alias. Integration Server continues making attempts, using each enabled proxy server alias (in an unspecified order) until the request is sent or all proxy servers have been tried but failed. After all proxy servers fail, Integration Server either sends the request to the remote server using a direct connection or throws an exception depending on the settings specified for the <code>watt.net.proxy.fallbackToDirectConnection</code> parameter.</p>
No	No	<p>Depends on the value specified for the <code>watt.net.proxy.useNonDefaultProxies</code> parameter.</p> <p>If <code>watt.net.proxy.useNonDefaultProxies</code> parameter is set to true, Integration Server makes outbound requests using each enabled proxy server alias (in an unspecified order) until the request is sent successfully or all proxy servers have been tried but failed. After all proxy servers fail, Integration Server sends the request to the remote server using a direct connection or throws an exception depending on the settings specified for the <code>watt.net.proxy.fallbackToDirectConnection</code> parameter.</p> <p>If no proxy server aliases exist for the specified protocol, Integration Server sends the request to the remote server using a direct connection or throws an exception depending on the settings specified for the</p>

Proxy Server Alias Specified?	Default Proxy Server Alias Exists?	Action taken by Integration Server
-------------------------------	------------------------------------	------------------------------------

watt.net.proxy.fallbackToDirectConnection parameter.

If the watt.net.proxy.useNonDefaultProxies parameter is set to false, Integration Server does not attempt to make outbound requests using the enabled proxy server aliases. Integration Server sends the request to the remote server using a direct connection.

Creating a Proxy Server Alias

Proxy server alias names must be unique across protocols. In other words, you cannot have proxy server aliases of the same name but of different protocols. For example, you can have only one proxy server alias named “myProxy” across the HTTP, HTTPS, FTP, or SOCKS protocols.

> To create a proxy server alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. Click **Create Proxy Server Alias**.
4. Under **Proxy Server Alias Properties**, provide information in the following fields:

For this parameter...	Specify...
Alias	The alias name to use for this host/port combination.
Host Name or IP Address	The host name or IP address of the proxy server.
Port Number	The port on which this proxy server listens for requests.
User Name (optional)	The user name Integration Server must use when accessing this proxy server.
Password (optional)	The password Integration Server must use to access this proxy server.
Protocol	The type of protocol (HTTP, HTTPS, FTP, or SOCKS) to use for the host/port combination.
Proxy Type	If you select FTP , additional parameters appear to help you configure the Proxy Type . For requests that the proxy server will

For this parameter...**Specify...**

be forwarding to an FTP server, you must specify the following information:

- The name of the FTP server
- The name of a user on the FTP server
- The password for that user.

The method you use to send this information to the FTP proxy server depends on the type of proxy server you have. Integration Server supports the following proxy server types:

0. No proxy

Do not use an FTP proxy server. This is the default.

1. ftp_user@ftp_host no proxy auth

Connect to the proxy server, but do not log into it. Then send the following:

```
USER ftp_user@ftp_host
PASS ftp_password
```

2. ftp_user@ftp_host proxy auth

Connect to the proxy server, and log into it with:

```
USER proxy_user
PASS proxy_password
```

Then send the following:

```
USER ftp_user@ftp_host
PASS ftp_password
```

3. site command

Connect to the proxy server, and log into it with:

```
USER proxy_user
PASS proxy_password
```

Then send the following:

```
SITE ftp_host
USER ftp_user
PASS ftp_password
```

4. open command

Connect to the proxy server, and log into it with:

For this parameter...**Specify...**

```
USER proxy_user
PASS proxy_password
```

Then send the following:

```
OPEN ftp_host
USER ftp_user
PASS ftp_password
```

5. ftp_user@proxy_user@ftp_host

Connect to the proxy server and log into it. Then send the following:

```
USER ftp_user@proxy_user@ftp_host
PASS ftp_password@proxy_password
```

6. proxy_user@ftp_host

Connect to the proxy server, and log into it with:

```
USER proxy_user@ftp_host
PASS proxy_password
```

Then send the following:

```
USER ftp_user
PASS ftp_password
```

7. ftp_user@ftp_host proxy_user

Connect to the proxy server, but do not log into it. Then send the following:

```
USER ftp_user@ftp_host proxy_user
PASS ftp_password
ACCT proxy_password
```

SOCKS Version

If you select **SOCKS** as the protocol, additional parameters appear to help you configure the SOCKS version. Select **SOCKS v4** or **SOCKS v5** based on what version of SOCKS protocol you would like to use to connect to the proxy server.

Note:

SOCKS protocol version 4 does not support authentication. Do not specify authentication credentials (i.e. username and password) if you select SOCKS protocol version 4.

Default

Whether this proxy server alias should be the default proxy server alias for its protocol type or not. Click **Yes** or **No**. Only one default proxy server alias can be set for each protocol type.

5. Click **Save Changes**.

Editing a Proxy Server Alias

If you need to make changes to the properties of a particular proxy server alias, you can edit it.

> To edit a proxy server alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. On the **Proxy Servers List** table, under **Alias** column, click on the name of the proxy server alias you wish to edit.

The **Edit** screen appears.

4. Modify the required **Proxy Server Alias Properties**.
5. Click **Save Changes**.

Disabling a Proxy Server Alias

Use the following procedure to disable a proxy server alias.

> To disable a proxy server alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. On the Proxy Servers main screen, click **Yes** under the **Enabled** column of the **Proxy Servers List**.

Integration Server displays a dialog box that prompts you to confirm your action.

4. Click **OK** to disable the proxy server alias.

Enabling a Proxy Server Alias

Use the following procedure to enable a proxy server alias.

> To enable a proxy server alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. On the Proxy Servers main screen, click **No** under the **Enabled** column of the **Proxy Servers List**.

Integration Server displays a dialog box that prompts you to verify your action.

4. Click **OK** to enable the proxy server alias.

Specifying a Default Proxy Server Alias

You can identify a default proxy server alias for each protocol type. Integration Server uses the default proxy server alias when one is not specified by the `pub.client:http`, `pub.client:ftp.login`, or `pub.client:ftp` services. Integration Server also uses the default proxy server alias when a consumer web service endpoint alias for the HTTP or HTTPS protocols does not specify a proxy server alias.

You can make any proxy server alias the default alias. However, there can be only one default proxy for a given protocol type.

> To select the default proxy server alias for a protocol

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. On the **Proxy Servers List** table, under **Alias** column, click on the name of the proxy server alias to use as the default.
4. Under **Proxy Server Alias Properties**, click **Yes** in the **Default** section to make this your new default proxy server.
5. Click **Save Changes**.

If another proxy server alias is already configured as the default proxy server alias for that protocol, Integration Server displays a dialog box that prompts you to confirm the change.

6. Click **OK** to change the default proxy server alias.

Deleting a Proxy Server Alias

If you no longer need a proxy server alias, you can delete it.

> To delete a proxy alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. In the **Proxy Servers List**, locate the alias you want to delete and click the **×** icon in the **Delete** column.

Integration Server displays a dialog box that prompts you to confirm your action.

4. Click **OK** to delete the selected proxy server alias.

Bypassing a Proxy Server

If you are using a proxy server for outbound HTTP, HTTPS, FTP, or SOCKS requests, you can optionally route selected requests directly to their targets, bypassing the proxy.

To do this, use the Integration Server Administrator to define a list of domains to which you want Integration Server to issue requests directly.

> To specify the proxy bypass addresses

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Proxy Servers**.
3. Click **Edit Proxy Bypass**.
4. In the **Addresses** field, type the fully qualified host and domain name of each server to which you want the Integration Server to issue requests directly. Type the host name and the domain name exactly as they appear in the URLs the server uses. To enter multiple names, separate each with commas.

You can use the asterisk (*) to identify several servers with similar names. The asterisk matches any number of characters. For example, if you want to bypass requests made to localhost, www.yahoo.com, home.microsoft.com, and all hosts whose names begin with NYC, you would type:

```
localhost,www.yahoo.com,home.microsoft.com, NYC*.*
```

5. Click **Save Changes**.

The proxy bypass addresses appear under **Proxy Bypass** on the Proxy Servers main screen.

Configuring Where the Integration Server Writes Logging, Status, and Other Information

Integration Server collects and stores data about the different areas of Integration Server functioning, including internal processing (scheduled jobs, Guaranteed Delivery, trigger joins), auditing and logging, central user management, document history, and process integrity. This data is stored in various database components that are identified to Integration Server through *functional aliases*. For information about the data stored in the different database components, and instructions on configuring functional aliases, see [“Connecting Integration Server to Database Components in an External RDBMS” on page 139](#).

Switching from the Embedded Database to an External RDBMS

If you installed your Integration Server with an embedded database, you can later switch to an external RDBMS.

➤ To switch from an embedded database to an external RDBMS

1. Navigate to the **Security > Certificates > Client Certificates** screen on the Integration Server Administrator, and make a note of the certificate mappings.
2. Create the IS Internal and Cross Reference database components and connect them to JDBC connection pools. For instructions, see [“Connecting Integration Server to Database Components in an External RDBMS” on page 139](#).

Note:

After you connect the database components to the JDBC connection pools, Integration Server will begin writing to the external RDBMS. You do not have to set any properties.

3. Run the migration utility `pub.scheduler:migrateTasksToJDBC` to migrate your scheduled tasks from the embedded database to the external RDBMS. See the *webMethods Integration Server Built-In Services Reference* for more instructions.

Note:

This service migrates scheduled tasks only; certificate mappings and run-time data stored in the embedded database will not be migrated.

4. Navigate to the **Security > Certificates > Client Certificates** screen on the Integration Server Administrator and re-specify your certificate mappings. See [“Importing a Certificate \(Client or CA Signing Certificate\) and Mapping It to a User” on page 519](#) for instructions.

Working with Extended Configuration Settings

There may be times when you want to view special server property settings. These properties are specified in the `server.cnf` file, however you can view them and edit them using the Integration

Server Administrator. Typically, you do not need to change these settings unless directed to by webMethods documentation or Software AG Global Support.

Important:

Typically, you will use the Integration Server Administrator to set properties in the server.cnf file, but there may be times when you need to edit the file directly with a text editor. *Before updating this file directly, be sure to shut down the Integration Server .*

> To view and edit extended configuration settings

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Extended**.

The server displays a screen that lists configuration properties specified in the server.cnf file.

3. By default, no properties are shown. If the properties you want to view are shown, skip this step. To select properties to be displayed, click **Show and Hide Keys**.

The server displays a list of all properties included in the server.cnf file (their values are not shown.) Select the box to the left of each property you want the server to display and click **Save Changes**. The server displays the **Extended Settings** screen again, this time with the selected properties and their values displayed.

4. To add, delete, or change a property setting, click **Edit Extended Settings** and type your changes.

Important:

Any change you make here will be reflected in the server.cnf file.

5. Click **Save Changes**.

Any properties you added will automatically display a check mark ✓ in the **Show and Hide Keys** list and will be displayed, with their values, in the **Extended Settings** list.

6. Restart the server for the changes to take effect.
 - a. In the upper right corner of any Integration Server Administrator screen, click **Shutdown and Restart**.
 - b. Select whether you want the server to wait before restarting or to restart immediately.
 - c. Click **Restart**.

Specifying Character Encoding

To ensure interpretability with other applications, Integration Server supports multiple forms of character encoding. The following table shows the default settings and the server properties that control them.

Action	Default Setting	Controlling Property
Reading and writing text files	Your JVM's file.encoding property	watt.server.fileEncoding
Reading text from and writing text to the network	UTF-8	watt.server.netEncoding

Important:

Consult with Software AG Global Support before changing these settings. The default settings are appropriate in most cases. Setting them incorrectly can cause unpredictable results.

Configuring the JVM

The configuration settings for the JVM in which Integration Server runs are specified by properties contained in the `custom_wrapper.conf`. These settings are given to the Java Service Wrapper when it launches the JVM. The procedures for updating the JVM configuration settings are provided in the following topics:

- [“Specifying the JDK or JRE for Integration Server” on page 129](#)
- [“Changing the JVM Heap Size Used by Integration Server” on page 130](#)
- [“Passing Java System Properties to Integration Server” on page 51](#)

For more information about the Java Service Wrapper, see *Software AG Infrastructure Administrator's Guide*.

Specifying the JDK or JRE for Integration Server

Integration Server must point to a JDK or JRE. By default, Integration Server points to the location of the JDK installed at the same time you installed Integration Server. If necessary, you can specify a different location.

Before you specify the location of Java for Integration Server, determine whether you need to specify the location of the JDK or the JRE. If you intend to use Designer to develop and compile Java services on Integration Server, specify the location of the JDK. If you will not be using this installation of Integration Server to compile Java services, you can specify the location of a JRE.

Important:

If you specify a different JDK or JRE, do *not* remove the JDK or JRE that Software AG Installer installed with Integration Server. The JDK and JRE installed with Integration Server are required to run the Software AG Uninstaller.

Note: Microservices Runtime does not use the Java Service Wrapper, the `wrapper.conf` file, or the `custom_wrapper.conf` file. To change the Java locations for Microservices Runtime, use a text editor to open `Integration Server_directory/bin/setenv.bat(sh)` and set the `JAVA_DIR` property to the JDK or JRE location.

> To specify the Java location for Integration Server

1. Open the `custom_wrapper.conf` file in a text editor. You can find the `wrapper.conf` file in the following location:

```
Software AG_directory \profiles\IS_instance_name\configuration
```

2. Add the `wrapper.java.command` property and set it so that it specifies the location of the JDK or JRE installation directory. For example:

```
wrapper.java.command=C:\SoftwareAG\jvm\jvm\bin\java
```

Note:

If your Integration Server runs on a UNIX system, you must also modify the library path property so that it points to the location of the JDK or JRE installation directory. Library path is represented by `LIBPATH` on AIX and `LD_LIBRARY_PATH` on all other flavors of Linux.

3. Save and close the file.
4. Restart Integration Server.

Note:

If you change the Java location and you use Integration Server to develop and compile Java services, you must also change the value of the `watt.server.compile` configuration parameter. For more information about `watt.server.compile`, see [“Server Configuration Parameters” on page 1017](#).

Changing the JVM Heap Size Used by Integration Server

The JVM heap or *on-heap* size indicates how much memory is allotted for server processes. At some point, you might need to increase the minimum and maximum heap size to ensure that the JVM that Integration Server uses does not run out of memory. You will want to consider the heap size when you configure your server to publish and subscribe to documents and when you configure an on-heap cache.

The heap size is controlled by the following Java properties specified in the `custom_wrapper.conf` file.

Property	Description
<code>wrapper.java.initmemory</code>	The minimum heap size. The default value is 256 MB.
<code>wrapper.java.maxmemory</code>	The maximum heap size. The default value is 1024 MB.

Your capacity planning and performance analysis should indicate whether you need to set higher maximum and minimum heap size values.

Note:Microservices Runtime does not use the Java Service Wrapper and the `custom_wrapper.conf` file. To change the memory settings for Microservices Runtime, use a text editor to set the `JAVA_MIN_MEM` and `JAVA_MAX_MEM` properties in: *Integration Server_directory* /bin/setenv.bat(sh).

➤ To change the heap size

1. Open the `custom_wrapper.conf` file in a text editor. You can find the `custom_wrapper.conf` file in the following location:

```
Software AG_directory \profiles\IS_instance_name\configuration
```

2. Set the `wrapper.java.initmemory` and `wrapper.java.maxmemory` parameters so that they specify the minimum and maximum heap size required by Integration Server. For example:

```
wrapper.java.initmemory=256
wrapper.java.maxmemory=512
```

3. Save and close the file.
4. Restart Integration Server.

Publishing and Retracting Information about Integration Server Assets

You can publish information or *metadata* to a shared library or registry that resides in CentraSite. By publishing this metadata, you make these assets available to other CentraSite users. You can also retract published metadata from CentraSite.

The metadata that you can publish are about Integration Server assets, Integration Server administrative assets, and Trading Networks (TN) document types. Integration Server administrative assets include Adapter connections, Broker connections, and JMS connection aliases. For a complete list of assets that you can publish, see *webMethods Service Development Help*.

You can use Software AG Designer to select the assets for which you want to publish or retract metadata to or from CentraSite. Designer publishes metadata using the credentials defined in the CentraSite connection in Designer.

You can also use the `pub.metadata.assets:publishPackages` service in the `WmAssetPublisher` package to publish metadata for packages and administrative assets. You can create a scheduled task that uses this service to publish metadata regularly. The `pub.metadata.assets:publishPackages` service uses the CentraSite credentials defined in Integration Server to publish metadata.

Note:

Before publishing metadata about Integration Server assets, you must configure Integration Server to connect to CentraSite. For instructions, see [“Configuring Integration Server to Connect to CentraSite” on page 132](#).

For detailed instructions on publishing and retracting metadata, see *webMethods Service Development Help*. For information about administering the shared library in CentraSite, see the CentraSite documentation.

Configuring Integration Server to Connect to CentraSite

Before you can publish metadata using the `pub.metadata.assets:publishPackages` service, you must configure Integration Server to connect to the shared library in CentraSite.

➤ To configure Integration Server to connect to CentraSite

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Metadata**.
3. Click **Edit Configuration**.
4. Under **Metadata Library Configuration**, provide information in the following fields:

For this parameter...	Specify...
IS Identifier	The name or the IP address of the machine on which this Integration Server is running. By default, Integration Server populates this field with the IP address of the machine. You can, however, enter a name here, which Integration Server will treat as an alias. In addition, Integration Server includes this name with the published metadata. You can use any name, but it cannot include the following illegal characters: #-&@^!%*:.\$/\ \ ` ; ~ +=)(}{][><" .
CentraSite URL	The CentraSite URL to which to publish metadata about Integration Server assets.
User Name	The name of the user account on CentraSite that will be used for publishing and retracting metadata.
Password	Password for User Name .

5. Click **Save Changes**.

You can test the configured connection to ensure that Integration Server can connect to CentraSite successfully using the details you provided. For instructions, see [“Configuring Integration Server to Connect to CentraSite” on page 132](#).

Testing the Connection to CentraSite

After you configure Integration Server to connect to CentraSite, you can test the connection to ensure that the specified host name, IP address, and library name identify a CentraSite library that is currently running.

› To test the connection to CentraSite

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Metadata**.
3. Click **Test Connection**.

Integration Server Administrator displays a status line that indicates whether the connection is successful or not. The status line is displayed at the top of the screen.

Setting a Port for Remote Client JMX Monitoring

Integration Server enables you to use JMX monitoring from a remote client. JMX monitoring is enabled by default, but you can set the JMX monitoring remote port in the `com.software.jmx.connector.pid-port.properties` file of the Integration Server you want to monitor, where *port* is the current JMX monitoring port. You cannot disable JMX monitoring.

Note:

A JMX port is not enabled by default for Microservices Runtime. To enable a JMX port for Microservices Runtime, modify *Integration Server_directory* /bin/setenv.bat(sh) to include `JMX_ENABLED=true` and `JMX_PORT=9192`. For more information, see *Developing Microservices with webMethods Microservices Runtime*.

› To set the port for remote client JMX monitoring

1. On the Integration Server that you want to monitor remotely, open the `com.software.jmx.connector.pid-port.properties` file in a text editor. You can find the `com.software.jmx.connector.pid-port.properties` file in the following location:

```
Software AG_directory
\profiles\IS_instance_name\configuration\com.softwareag.platform.config.propsloader
```

2. Specify the port number of the JMX port in the `port` property. This is 8075 by default.
3. Restart Integration Server.

Configuring Integration Server to Accept a Debug Connection During Startup

Using Integration Server, you can connect a debugging tool to remotely debug Java services running on Integration Server. You can specify the port on which Integration Server listens for the debugger to attach.

For information about using Software AG Designer to debug Java services that run on Integration Server, see *Software AG Designer Online Help*.

Note:

A JMX port is not enabled by default for Microservices Runtime. To enable a JMX port for Microservices Runtime, modify *Integration Server_directory/bin/setenv.bat(sh)* to include `JMX_ENABLED=true` and `JMX_PORT=9192`.

> To configure Integration Server to accept a debug connection during startup

1. Shut down Integration Server.
2. If you need to change the port number, perform the following:
 - a. Open the `startDebugMode.bat` file in a text editor. You can find the `startDebugMode.bat` file in the following location:


```
Software AG_directory \profiles\IS_instance_name\bin
```
 - b. Locate and change the `DEBUG_PORT` property to specify the port on which the server should listen for the debugger to attach. The default is 10033.
 - c. Locate `SUSPEND_MODE` property and set it to `y` if you want the JVM to wait for a debugger to attach.
 - d. Save your changes and close the `startDebugMode.bat` file.
3. If Integration Server and the debugging tool are on different machines and you require a firewall port, open a firewall port for the debug port.
4. Run `startDebugMode.bat`.

Integration Server logs the following on your console:

```
"Debug enabled (portNumber)"
Listening for transport dt_socket at address: portNumber
```

Integration Server restarts.

Using CORS with Integration Server

CORS (Cross-Origin Resource Sharing) is a specification that provides the standards for user agents (such as web browsers) to allow client-side web applications running in one origin to access resources from another origin. For security reasons, cross-origin requests initiated by scripts are usually restricted by user agents; however, user agents can use CORS to allow and define these cross-origin communications.

How Integration Server Handles CORS Requests

Integration Server provides a set of extended settings to enable and configure processing of CORS requests. A CORS request contains a set of HTTP headers that can be used to identify the request and communicate with Integration Server. When support for CORS is enabled, Integration Server checks all incoming requests for these headers. If a request contains CORS headers, Integration Server attempts to validate the request. Integration Server processes valid requests and includes CORS response headers in the response.

Note:

To use CORS, both the server and the client must communicate with the CORS headers as defined by the CORS standard.

Configuring Integration Server to Accept CORS Requests

Use the following procedure to configure your Integration Server to accept and process CORS requests.

➤ **To enable CORS processing on Integration Server**

1. In the **Settings** menu of the Navigation panel, click **Extended**.
2. Click **Edit Extended Settings**.
3. Type `watt.server.cors.enabled=true`.
4. Specify the URIs from which Integration Server is to allow cross-origin requests to access resources by doing one of the following:

- To specify the URIs as the value of `watt.server.cors.allowedOrigin`, do the following:

```
watt.server.cors.allowedOrigins=<protocol>://<hostname>
```

or

```
watt.server.cors.allowedOrigins=<protocol>://<hostname>:<port>
```

Where *<protocol>* is either HTTP or HTTPS, *<hostname>* is the IP address or name of the machine, and *<port>* is the port number.

Note:

The host name and IP address cannot be used interchangeably. If you specify a host name, and a cross-origin request is received that uses the corresponding IP address or vice versa, Integration Server will reject the request.

- To specify a text file that contains allow of the allowed URIs, set the value to:

`file:path\filename`

For example: `watt.server.cors.allowedOrigins=file:c:\cors\allowedOriginsList.txt`

Note:

The file must use the same syntax as is required for URIs specified as the value for `watt.sever.cors.allowedOrigins` with the difference that each line must be an allowed origin server URI or a regular expression.

The values are case-sensitive. Use a space or comma delimiter to specify multiple values. You can use an asterisk (*) to indicate that any URI or origin is allowed. You can also use regular expressions in the comma-separated list of allowed origin servers. For more information about setting the `watt.server.cors.allowedOrigins` service configuration parameter, see [“Server Configuration Parameters” on page 1017](#).

5. Optionally, set any of the following parameters for CORS processing:

If you want to...	Type...
Specify values that Integration Server can include with the CORS Access-Control-Expose-Headers header in response to a CORS request.	<pre>watt.server.cors.exposedHeaders= <value1,value2,value3></pre> <p>Where <i>value#</i> is a response header that applications can access. Examine your client-side code to determine which response headers, if any, are retrieved by the client and therefore need to be exposed.</p>
Specify the host and port on which clients can send cross-origin requests to Integration Server.	<pre>watt.server.cors.host=<hostname>:<port></pre>
Specify the amount of time in seconds a user agent is allowed to cache the results of a preflight request.	<pre>watt.server.cors.maxAge= <number of seconds></pre>
Have Integration Server set the CORS Access-Control-Allow-Credentials header in response to all CORS requests.	<pre>watt.server.cors.supportsCredentials=true</pre>
Specify the request headers Integration Server will allow in cross-origin requests. Integration Server includes these headers as the value of the Access-Control-Allow-Headers	<pre>watt.server.cors.supportedHeaders= <header1,header2,header3></pre> <p>Where <i>header#</i> is a header that a client can include in an actual request. Examine your server-side</p>

If you want to...	Type...
response header when replying to a pre-flight request.	code to determine which request headers, if any, are read by your server application and need to be explicitly allowed.
Specify the HTTP methods Integration Server will allow in cross-origin requests.	<pre>watt.server.cors.supportedMethods= <method1,method2,method3></pre> <p>Possible values are OPTIONS, HEAD, GET, POST, PUT, PATCH, and DELETE. Integration Server accepts any of these values by default.</p>

See [“Server Configuration Parameters” on page 1017](#) for important usage information and detailed descriptions of these parameters.

6. Click **Save Changes**.
7. If you used a text file as the value of `watt.server.cors.allowedOrigins`, invoke the following URL to execute an internal service that causes the referenced text file to be loaded into Integration Server: `http://host:port/invoke/wm.server.admin:refreshAllowedOrigins`

Where *host* and *port* are the host and port of Integration Server

8 Connecting Integration Server to Database Components in an External RDBMS

■ Concepts	140
■ Pointing Functions at Connection Pools	140
■ Installing the MySQL Community Edition Database Driver	141
■ Installing the Tibero JDBC Driver	141
■ Creating a Driver Alias Definition	142
■ Creating a Connection Pool	142

Concepts

You connect Integration Server to database components in an external RDBMS as follows:

- Define *JDBC database connection pools*. Connection pools specify the parameters for connecting Integration Server to database servers that host database components. At run time, Integration Server creates a separate instance of the connection pool for each database component.
- Direct *functions* to write to database components by pointing each function at the connection pool for the database component. For example, point the ISCoreAudit Log function at the connection pool for the ISCoreAudit database component, the Xref function at the connection pool for the CrossReference database component, and so on. Integration Server provides a predefined function for each type of data that can be written to a database component.

When you choose to use an external database during Integration Server installation, the Software AG Installer asks you to supply database connection parameters. When you first start Integration Server, Integration Server creates one JDBC connection pool from the supplied parameters and configures the predefined functions to write to their database components using that pool.

The following table identifies the predefined functions and their database components that are included with Integration Server.

Function	Database Component
ISInternal	ISInternal and DistributedLocking
ISCoreAudit	ISCoreAuditLog
Xref	CrossReference
DocumentHistory	DocumentHistory
ProcessAudit	ProcessAudit
Process Engine	ProcessEngine

You cannot add or delete functions, but you can create a connection pool and point a function to the new pool.

Pointing Functions at Connection Pools

You can refer to the following steps to point a

➤ To point a function at a connection pool

1. In Integration Server Administrator, go to **Settings > JDBC Pools**.
2. In the **Functional Alias Definitions** area, click **Edit** for the function.

3. In the **Associated Pool Alias** field, identify the connection pool at which to point the function.
4. If the **Fail-Fast Mode Enabled** option is available, choose whether to enable or disable.

Transient errors caused by an unavailable database can prevent a connection pool from connecting to its database. You can configure a function to enter *fail-fast mode* to handle this situation. In fail fast mode, all attempts by the function to get a database connection immediately return a `SQLException`; this saves time by preventing retry attempts. When database connectivity is restored, the function exits fail-fast mode and returns to normal operation. Fail-fast mode can improve performance when you are using synchronous audit logging (see the).

Note:

Fail-fast cannot be used with the `CentralUsers` functional alias definition. Consequently, the **Fail-Fast Mode Enabled** and **Currently In Fail-Fast** options do not appear when editing the `CentralUsers` functional alias.

5. Click **Save Settings**.
6. In the **Functional Alias Definitions** area, initialize the pool by clicking **Restart** for the function.
7. Make sure Integration Server can connect to the database by clicking ► for the function.
8. Restart Integration Server.

Installing the MySQL Community Edition Database Driver

Refer to the simple step below to install the MySQL Community Edition database driver.

1. Download the MySQL Community Edition database driver.
2. Extract the downloaded file and copy the driver JAR to: *Software AG_directory* \IntegrationServer\lib\jars\custom.

Installing the Tibero JDBC Driver

Integration Server supports the use of the Tibero JDBC driver to connect to a Tibero database instance.

Note:

Support for using Integration Server with the Tibero JDBC driver to connect to a Tibero database is provided as part of PIE-64402, which is included in IS_10.5_Core_Fix6.

► To install the Tibero JDBC driver

- Download the Tibero JDBC Driver jar file and place it in the following directory:

Software AG_directory \IntegrationServer\lib\jars\custom

Creating a Driver Alias Definition

A driver alias definition identifies a database driver to Integration Server. While Integration Server comes with some driver aliases already defined, you may need to create a driver alias definition.

> To create a driver alias definition

1. Open the Integration Server Administrator if it is not already open.
2. Go to **Settings > JDBC Pools > Create a Driver Alias Definition**.
3. Provide the following information for the driver alias.

Parameter	Description
Alias Name	Alias name to use for the driver. The name can be any length and contain any characters.
Alias Description	Brief description of the driver alias.
Driver Class Name	Name of the Java class for the JDBC driver. For the Tibero JDBC Driver, set the class name to: <code>com.tmax.tibero.jdbc.TbDriver</code>

4. Click **Save Settings**.

Creating a Connection Pool

You can create a connection pool by copying an existing pool using **Copy a Pool Alias Definition** or by specifying new field values as instructed in this section.

> To create a connection pool by specifying new field values

1. In Integration Server Administrator, go to **Settings > JDBC Pools**, click **Create a Pool Alias Definition**, and complete the fields.

Parameter	Specify
Alias Name	Name to use for the connection pool. The name can include any characters that are valid for a file name in your operating system.
Alias Description	Brief description of the pool.

Parameter	Specify
Associated Driver Alias	Database driver for Integration Server to use to connect to the pool.
Database URL	<p>URL for the database server.</p> <ul style="list-style-type: none"> ■ Use the DataDirect Connect connection option <code>MaxPooledStatements=35</code> on all database URLs except those for Trading Networks. This connection option improves performance by caching prepared statements. (Trading Networks caches its prepared statements using its own pooling mechanism). ■ For DB2, if Integration Server will connect to a schema other than the default schema for the specified database user, you must specify these connection options in the URL: <pre> ;AlternateId=schema;"InitializationString=(SET CURRENT PATH=current_path,schema)"; MaxPooledStatements=35 </pre> <p>AlternateID is the name of the default schema that is used to qualify unqualified database objects in dynamically prepared SQL statements.</p> ■ For MySQL Community Edition, you must specify connection options for the <code>relaxAutoCommit</code>, <code>useLegacyDatetimeCode</code>, and <code>serverTimezone</code> parameters. For example, you can provide the connection options as follows: <pre> jdbc:mysql://<server>:<3306 port>/databaseName? relaxAutoCommit=true& useLegacyDatetimeCode=false& serverTime </pre> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: MySQL Community Edition 8.x does not support the <code>relaxAutoCommit</code> parameter in the URL because it is deprecated and Integration Server does not commit the transactions automatically to MySQL Community Edition 8.x.</p> </div> <p>For MySQL Community Edition, you must specify the <code>useCursorFetch</code> parameter in the URL to prevent the return of all the query results in a single batch. The MySQL JDBC driver used with the MySQL Community Edition does not honor the cursor fetch size when returning the results of an SQL query. Instead of returning the JDBC-configured number of rows for each cursor fetch to Integration Server, the MySQL JDBC driver returns all the rows for the query in a single batch. This can result in the exhaustion of the available heap memory for Integration Server. Any JDBC connection pool that connects to MySQL Community Edition must specify</p>

Parameter	<p>Specify</p> <p>the following connection option in the Database URL: useCursorFetch=true</p> <ul style="list-style-type: none"> ■ For Tiberio, specify: jdbc:tiberio:thin:@<server>:<8629 port>:databaseName 						
Diagnostics	<p>DataDirect diagnostic feature for DataDirect Connect JDBC drivers used by Integration Server to interact with an external database.</p> <table border="1"> <thead> <tr> <th style="text-align: left;">Option</th> <th style="text-align: left;">Description</th> </tr> </thead> <tbody> <tr> <td>Spy</td> <td>Enables the DataDirect Spy diagnostic feature for DataDirect Connect JDBC drivers. DataDirect Spy logs JDBC calls and SQL statement interactions between Integration Server and an external RDBMS.</td> </tr> <tr> <td>Snoop</td> <td>Enables the DataDirect Snoop tool for DataDirect Connect JDBC drivers. The DataDirect Snoop tool logs network packets between Integration Server and an external RDBMS. You can use the resulting log file for tracing and diagnostic purposes.</td> </tr> </tbody> </table>	Option	Description	Spy	Enables the DataDirect Spy diagnostic feature for DataDirect Connect JDBC drivers. DataDirect Spy logs JDBC calls and SQL statement interactions between Integration Server and an external RDBMS.	Snoop	Enables the DataDirect Snoop tool for DataDirect Connect JDBC drivers. The DataDirect Snoop tool logs network packets between Integration Server and an external RDBMS. You can use the resulting log file for tracing and diagnostic purposes.
Option	Description						
Spy	Enables the DataDirect Spy diagnostic feature for DataDirect Connect JDBC drivers. DataDirect Spy logs JDBC calls and SQL statement interactions between Integration Server and an external RDBMS.						
Snoop	Enables the DataDirect Snoop tool for DataDirect Connect JDBC drivers. The DataDirect Snoop tool logs network packets between Integration Server and an external RDBMS. You can use the resulting log file for tracing and diagnostic purposes.						
Spy Attributes	<p>Name and location of the log file where Integration Server will log diagnostic data collected by the DataDirect Spy diagnostic feature. This value also defines DataDirect Spy attributes. The default value is:</p> <pre>SpyAttributes=(log=(file)/logs/spy/<alias_name>.log; logTName=yes;timestamp=yes)</pre> <p>Where <i><alias_name></i> is the name of the JDBC connection pool alias.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: Typically, you do not need to change the default value. However, you can modify the value if the attributes do not meet the needs of your system. Be aware that the diagnostic tool collects data from the <i>Integration Server_directory/instances/instance_name/logs/spy</i> directory. If you change the log file location, the diagnostic utility might not import the data logged by DataDirect Spy. For more information about using the diagnostic utility, see “Using the Diagnostic Utility” on page 939. For more information about setting DataDirect Spy attributes, consult the documentation on the DataDirect website.</p> </div> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: MySQL Community Edition does not support this option.</p> </div>						

Parameter	Specify
Snoop Logging Parameters	<p>Name and location of the log file where Integration Server will log diagnostic data collected by the DataDirect Snoop tool. This value also defines DataDirect Snoop tool attributes. The default value is:</p> <pre>ddtdbg.ProtocolTraceEnable=true;ddtdbg.ProtocolTraceMaxline=16;ddtdbg.ProtocolTraceLocation=/logs/snoop/<i>alias_name</i>.log;ddtdbg.ProtocolTraceShowTime=true</pre> <p>Where <i>alias_name</i> is the name of the JDBC connection pool alias.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: For DB2, include the following command at the end of the value:</p> <pre>ddtdbg.ProtocolTraceEBCDIC=true</pre> </div> <p>Typically, you do not need to change the default value. However, you can modify the value if the attributes do not meet the needs of your system. Be aware that the diagnostic tool collects data from the <i>Integration Server_directory</i> /instances/instance_name/logs/snoop directory. If you change the log file location, the diagnostic utility might not import the data logged by the DataDirect Snoop tool. For more information about using the diagnostic utility, see “Using the Diagnostic Utility” on page 939. For more information about setting the DataDirect Snoop tool attributes, consult the documentation on the DataDirect website.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: MySQL Community Edition does not support this option.</p> </div>
User ID	Database user for Integration Server to use to connect to the database.
Password	Password for Integration Server to use to connect to the database. If no password is required, leave this field blank.
Minimum Connections	<p>Minimum number of connections the pool must keep open at all times.</p> <p>If multiple database components use this pool, each pool instance keeps the specified number of connections open. For example, if the ISCoreAuditLog and the DocumentHistory database components both use this pool, and you specify keeping at least 3 connections open, the pool keeps a total of 6 connections open - 3 for the ISCoreAuditLog pool instance and 3 for the DocumentHistory pool instance.</p>

Parameter	Specify
Maximum Connections	<p>If your logging volume has sudden spikes, you can improve performance by making sure the connections needed to handle the increased volume open quickly. You can minimize connection startup time during spikes by setting this value higher, so that more connections remain open at all times.</p> <p>Maximum number of connections the pool can have open at one time. When the number of connection requests reaches this value, Integration Server blocks the requests.</p> <p>Calculate this value as part of the total possible number of connections that could be opened simultaneously by all functional aliases and applications that write to the database. Make sure the total number does not exceed the database's connection limit. If one of the applications opens more connections than the database allows, the database will reject subsequent requests for connections from any application.</p> <p>To continue the example described for Minimum Connections, if Trading Networks also writes to the database and has a pool that could open up to 5 connections, you could specify only 17 as the maximum number of connections for the current pool. The ISCoreAuditLog pool instance could use up to 17 connections, and the DocumentHistory pool instance could use the remaining 5 connections.</p>
Available Connections Warning Threshold	<p>Number of connections, expressed as a percentage of Maximum Connections, that should be available in the pool at all times.</p> <p>When the number of connections falls to or below this number, Integration Server logs a message to the server log. If the number of connections later rises above this number, Integration Server logs another message to the server log stating that the connection pool threshold has been cleared. To disable this threshold, set the value to 0.</p>
Waiting Thread Threshold Count	<p>Maximum number of requests for connection that can be waiting at one time.</p> <p>When this number is exceeded, Integration Server logs a message to the server log and starts a 5-minute interval timer. If the number of requests still exceeds this number at the end of the interval, Integration Server logs another message to the server log. To disable this threshold, set the value to 0.</p>
Idle Timeout	<p>Length of time, in milliseconds, the pool can keep an unused connection open. After the specified period of time elapses, the pool closes unused connections that are not needed to satisfy the</p>

- | Parameter | Specify |
|-----------|--|
| | Minimum Connections value. The default expiration time is 60000 milliseconds. |
2. Make sure Integration Server can connect to the database by clicking **Test Connection**.
 3. Click **Save Settings**.
 4. Check the values for the connection pool for the ISCoreAudit database component. If the database user specified in the **User ID** field is not the database user that created the ISCoreAudit database component, set the `watt.server.audit.schemaName` property to the name of the schema that contains the ISCoreAudit database component (see [“Server Configuration Parameters”](#) on page 1017).

9 Configuring Ports

■ About Ports	151
■ Considerations for Adding Ports	154
■ Adding an HTTP Port	155
■ Adding an HTTPS Port	162
■ About File Polling Ports	169
■ Adding an FTPS Port	175
■ Adding an FTP Port	179
■ Adding an E-Mail Port	181
■ Adding an HTTP Diagnostic Port	189
■ Adding an HTTPS Diagnostic Port	194
■ Suspending an HTTP/HTTPS Port	202
■ Resuming an HTTP/HTTPS Port	202
■ Testing for HTTPS Requests	203
■ Using an FTP/FTPS Port Range	203
■ About the Primary Port	204
■ Deleting a Port	205
■ Editing a Port	206
■ About Enabling/Disabling a Port	206
■ Configuring How Ports Handle Client Certificates	207

- Adding a Security Provider 208
- Configuring the Allowed Protocols for JSSE per Port 209
- Disabling TLS Renegotiation 210
- Designating an HTTP/S Port as Stateless 211

About Ports

Integration Server listens for requests on ports that you specify. Each port is associated with a specific type of protocol: HTTP, HTTPS, FTP, FTPS, WebSockets, e-mail or file polling. In addition to these port types, Integration Server also provides a diagnostic port, a quiesce port, and two ports used by webMethods Enterprise Gateway.

Available Port Types

The following table describes the port types that you can configure:

Use this port type	To	Refer to
HTTP	Submit unsecured requests to the server.	“Adding an HTTP Port” on page 155
HTTPS	Submit requests to the server using SSL encryption.	“Adding an HTTPS Port” on page 162
FTP	Move files to and from the server.	“Adding an FTP Port” on page 179
FTPS	Move files to and from the server using SSL encryption.	“Adding an FTPS Port” on page 175
E-mail	Receive requests through an e-mail server, such as POP3 or IMAP.	“Adding an E-Mail Port” on page 181
File polling	Monitor the file system for the arrival of new files and perform special processing upon arrival.	“Adding a File Polling Port” on page 170
Diagnostic	Access Integration Server Administrator when the server becomes unresponsive.	“Adding an HTTP Diagnostic Port” on page 189
Quiesce	Enter and exit quiesce mode for server maintenance.	“Quiescing the Server for Maintenance” on page 929
Enterprise Gateway Server	Listen for requests from external clients and maintain the connection to the Internal Server in a webMethods Enterprise Gateway configuration.	“Configuring webMethods Enterprise Gateway ” on page 557
Internal Server	Connect the Internal Server to the Enterprise Gateway Server in a webMethods Enterprise Gateway configuration.	“Configuring webMethods Enterprise Gateway ” on page 557

Use this port type	To	Refer to
webMethods/WebSockets	Submit requests for services using WebSocket protocol.	“Configuring WebSockets” on page 809

Default Ports

The default Integration Server instance has the following pre-configured ports:

Alias	Protocol	Port Type	Port Number
DefaultPrimary	HTTP	Primary	5555
DefaultDiagnostic	HTTP	Diagnostic	9999

Note:

The alias assigned to a default port on Integration Server cannot be changed. If you want to use a different alias for the port, protocol, and port type used by a default port, delete the port and recreate it with the new alias. To use a different alias for the DefaultPrimary port, you must designate another port as the primary port before deleting the DefaultPrimary port and recreating it with a new alias.

If you are running multiple Integration Servers on the same machine, the primary and diagnostic port numbers on each server must be unique. For more information about running multiple Integration Server instances, see [“Running Multiple Integration Server Instances” on page 63](#).

About the Port Alias

When you create a port, you specify an alias for the port. The port alias serves as a unique identifier for the port. Often, the alias is a short descriptive name that identifies the purpose of the port. For example, you might want the port alias to identify the primary function of the port, such as AdminPort, WebServicePort, or projectNamePort.

When using Software AG Command Central, the port alias can make it easier to manage, configure, and compare ports. For example, if you assigned all of the administration ports in a group of Integration Servers the same name or similar names, you can more easily identify and therefore more easily compare the administration ports.

A port alias must:

- Be unique across the Integration Server.
- Be between 1 and 255 characters in length.
- Include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).

Once the port is created, the alias cannot be changed. The default ports provided with Integration Server have pre-defined alias names as specified in [“Default Ports” on page 152](#).

When you upgrade to Integration Server 9.5 or later from a previous version that did not use port aliases, Integration Server assigns each port, with the exception of e-mail ports and file polling ports, an alias using the following naming convention:

protocolListener_portNumber_hostName_packageName

Where *protocol* is the protocol specified for the port, *portNumber* is the number assigned to the port, *hostName* is the host name specified for the port, and *packageName* is the package with which the port is associated.

For e-mail ports, Integration Server uses the following naming convention for the port alias:

EMailListener_userName_hostName_packageName

Where *userName* is the user name specified for the port, *hostName* is the host name specified for the port, and *packageName* is the package with which the port is associated.

For file polling ports, Integration Server uses the following naming convention for the port alias:

FilePollingListener_monitoringDirectory@contentType

Where *monitoringDirectory* is the directory that Integration Server monitors for new files, and *contentType* is the content type for the files the port processes. If the file polling port does not specify a content type, Integration Server omits *contentType* from the port alias.

Integration Server uses the naming conventions described above to create an alias for a port deployed from an earlier version of Integration Server and for a port installed with a package created on an earlier version of Integration Server.

Note:

The alias that Integration Server assigns to a port cannot be changed.

Package Associations for Ports

All ports are associated with a package. By default, ports are associated with the WmRoot package. You can associate all port types except the diagnostic port with an application package. By doing so, when you replicate the package, the package continues to use a port with the same number on the new server. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.

When the package to which a port is assigned is disabled, the port is disabled as well. The port is re-enabled when the package activates successfully. While the port is disabled, access to services or other resources through those ports will fail until the port is successfully enabled. If you plan to use hot deployment for packages, you may want to avoid associating ports with packages that will be updated via hot deployment. This will avoid the situation where the port and its resources are not available when Integration Server unloads the package during hot deployment.

Important:

Be careful when setting up a port that is associated with a package. When copied to the target server, the new port might decrease security on that system. For example, suppose you replicate a package that is associated with an HTTP port at 5556. The replication process creates an HTTP

port at 5556 on the target server. If the target server normally uses only HTTPS ports because of their greater security, then the new port presents a possible security hole on that server.

Considerations for Adding Ports

You can configure one or more additional ports. You can associate an HTTP, HTTPS, FTP, FTPS, e-mail, or file polling protocol with the additional ports.

Important:

If you are running multiple Integration Server instances on the same host machine, the ports on each server must have a unique port number.

Reasons to Add Additional Ports

You might add additional ports for the following reasons:

- If you have applications that require a specific port number.
- If you want to support multiple types of listening protocols.
- If you want to open several ports for the same protocol.
- If you want to deploy your server in a webMethods Enterprise Gateway configuration, in which an Enterprise Gateway Server sits in your DMZ and intercepts requests before passing them to the server behind your inner firewall. For instructions on adding Enterprise Gateway ports, see [“Configuring webMethods Enterprise Gateway” on page 557](#).

Considerations for Configuring Ports

The following are prerequisites and considerations for configuring ports.

AS/400 Considerations

The *port queue* is the number of outstanding inbound connections that are queued in a TCP/IP stack. If your server runs on AS/400, you should limit the size of the port queue by adding the line “watt.server.portQueue=511” to the server property settings.

For instructions on adding server property settings, see [“Working with Extended Configuration Settings” on page 127](#).

Bind Addresses

When you add a port, you can specify an IP address to bind to the port. Specify a bind address in the following situations:

- Your machine has multiple IP addresses, and you want the port to use a specific address.

- You want multiple ports to use the same port number. In this case, each port must specify a different bind address. For example, if port 7777 is associated with a bind address, you cannot add another port 7777 that does not specify a bind address.

If you do not specify a bind address, the port listens to all network interfaces.

Prerequisites to Configuring a Port for SSL

Before configuring an HTTPS, FTPS, or e-mail port, you must configure the server to use SSL and obtain the certificates that the server uses to validate client certificates. In addition, for two-way SSL authentication (server also authenticates client), the certificate for the partner application must have an Integration Server certificate mapping.

- **Configure the server to use SSL.** For information about configuring the server to use SSL, see [“Configuring Integration Server for Secure Communication” on page 457](#).
- **Obtain CA certificates.** These are the trusted root certificates that the server uses to validate client certificates. One way to obtain these certificates is to extract them from a web browser. Most web browsers that support SSL are shipped with the certificates of well-known certificate authorities. Make sure the certificates are in DER format; if not, convert them to DER format using your certificate management tool (such as Java *keytool*).
- **Configure a certificate mapping for the partner application or resource.** If the partner application or resource that exchange information with Integration Server must be authenticated using SSL, then that partner or resource must have a certificate mapping. These certificates should be stored in the truststore and an alias created for the truststore prior to using them with Integration Server. For information, see [“HTTPS Ports” on page 521](#).

Port Usage and Security

For security reasons, by default, all ports except 5555 are configured to deny access to all services except services specified in an allow list. You might need to perform additional steps to make more services available through the port. See [“Configuring How Ports Handle Client Certificates” on page 207](#)

Note:

If your Integration Server runs on a UNIX system, using a port number below 1024 requires that the server run as “root.” For security reasons, Software AG discourages this practice. Instead, run your Integration Server using an unprivileged user ID on a high number port (for example 1024 or above) and use the port remapping capabilities present in most firewalls to move requests to the higher numbered ports.

Adding an HTTP Port

To add an HTTP port, complete the following steps.

➤ To add an HTTP port

1. Open Integration Server Administrator if it is not already open.

2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Add Port**.
4. In the **Add Port** area of the screen, select **webMethods /HTTP**.
5. Click **Submit**. Integration Server Administrator displays a screen requesting information about the port. Provide the following information:

For this parameter...	Specify...
Enable	Indicate whether to enable (Yes) or disable (No) this HTTP listener.
Port	The number you want to use for the port. Select a number that is not already in use on this host machine. Important: If you are running multiple Integration Servers on the same host machine, make sure the port numbers used on each server are unique.
Alias	An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.
Package Name	The package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port. If you replicate this package, the Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.
Bind Address (optional)	IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.
Backlog	The number of requests that can remain in the queue for an enabled port before Integration Server begins rejecting requests. The default is 200. The maximum value is 65535.

Note:

For this parameter...	Specify...
	This parameter does not apply to disabled ports. Integration Server refuses requests sent to disabled ports.
Keep Alive Timeout	When to close the connection if the server has not received a request from the client within this timeout value (in milliseconds); or when to close the connection if the client has explicitly placed a close request with the server.
Threadpool	<p>Whether the listener will use this pool exclusively for dispatching requests. The existing Integration Server thread pool is a global thread pool. If there is a very high load on this resource, the user may have to wait for the global thread pool to process his request. However, with the private thread pool option enabled, requests coming into this port will not have to compete with other server functions for threads.</p> <p>To set up a private thread pool for requests coming to this port, click Enable. You can change or accept the default settings given below:</p> <p>Threadpool Min refers to the minimum number of threads for this private threadpool. The default is 1.</p> <p>Threadpool Max refers to the maximum number of threads for this private thread pool. The default is 5.</p> <p>Threadpool Priority refers to the Java thread priority. The default is 5.</p> <p>Important: Use this setting with extreme care because it will affect server performance and throughput.</p> <p>If you do not need to use the Threadpool feature, click Disable.</p> <p>When you view the port's details, the server reports the total number of private threadpool threads currently in use for the port.</p>

6. Under **Security Configuration**, enter the following information:

For this parameter...	Specify...		
Client Authentication	The type of client authentication you want Integration Server to perform for requests that arrive on this HTTP port. Select one of the following:		
	<table border="1"> <thead> <tr> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> </tbody> </table>	Option	Description
Option	Description		

For this parameter...	Specify...
Username/Password	Integration Server prompts the client for a user ID and password.
Digest	<p>Integration Server uses password digest to authenticate all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p> <p>A port that is configured to use password digest for authentication of client requests will process a request from a user only if the user is configured to allow password digest for authentication. For more information about configuring a user for digest authentication, see “Adding User Accounts” on page 86.</p>
Request Kerberos Ticket	<p>Integration Server looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server uses user name and password for basic authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p>
Require Kerberos Ticket	<p>Integration Server looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server fails</p>

For this parameter...	Specify...
Kerberos Properties (Optional)	<p>the authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p> <p>Kerberos properties are used to enable Kerberos authentication by providing Kerberos-related details that will be used for handling service requests that come with a Kerberos ticket. For information on configuring Kerberos authentication, see “Kerberos Authentication” on page 517.</p>
JAAS Context	<p>Specify the custom JAAS context used for Kerberos authentication.</p> <p>In the following example, JAAS Context is KerberosClient:</p> <pre data-bbox="982 1003 1458 1333">KerberosClient { com.sun.security.auth.module. Krb5LoginModule required useKeyTab=true keyTab=alice.keytab; };</pre>
Principal	<p>Specify the name of the principal to use for Kerberos authentication.</p>
Principal Password	<p>Specify the password for the principal that is used to authenticate the principal to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for</p>

For this parameter...	Specify...
	<p>authorization. The passwords may be encrypted using different encryption algorithms.</p> <p>If the JAAS login context contains <code>useKeyTab=false</code>, you must specify the principal password.</p>
	<p>Retype Principal Password Re-enter the principal password.</p>
	<p>Service Principal Name Format Displays username, which indicates that the principal name of the service is represented as a named user defined in the LDAP or central user directory used for authentication to the KDC.</p>
	<p>Service Principal Name Specify the name of the principal used with the service that the Kerberos client wants to access. Specify the Service Principal Name in the following format:</p> <p><i>principal-name.instance-name@realm-name</i></p>

- Click **Save Changes**.
- On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

- On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Advanced Controls for HTTP Ports

By default, Integration Server accepts port connections requests as soon as it receives them. This can be a problem if the port receives multiple requests simultaneously and does not have the resources to handle them. You can handle this by specifying a delay value using the Advanced Controls screen. With a delay value in place, Integration Server will wait the specified number of milliseconds before accepting a connection request on this port. The Advanced Controls screen provides you the capability to control the rate at which the listener accepts connections, over the size of the private thread pool, if it was enabled.

Editing Advanced Controls

To edit advanced controls, complete the following steps.

➤ To edit advanced controls

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.

You will see a **Port List** table in the main area of the screen.

3. In the **Advanced** column of this table, click **Edit**.

Integration Server displays a screen requesting information about Listener and Private Thread Pool Controls. Note that the Diagnostic HTTP Listener State and Private Threadpool areas of the screen are already populated with predefined values.

4. Enter the following information:

For this parameter...	Specify...
Listener controls	<p>The type of controls you want to set, to manage the rate at which the listener accepts connections and other controls when the private thread pool is enabled.</p> <p>Suspend. Stops the listener from accepting any more connections and subsequently dispatching any more requests.</p> <p>Increase By. Increases the time that the listener will wait before accepting new client connections.</p> <p>Decrease By. Decreases the time that the listener will wait before accepting new client connections.</p> <p>Set To (Delay ms). Sets the delay time interval in milliseconds.</p>
Private Thread Pool Controls	<p>The type of thread pool control you want, in order to avoid the need for your port to compete with other server functions when Integration Server is handling multiple connections.</p> <p>Increase By. Increases the number of threads in the private thread pool.</p> <p>Decrease By. Decreases the number of threads in the private thread pool.</p> <p>Set To (Threads). Sets the number of threads in the private thread pool.</p>

- Click **Apply** to accept your changes. Else, click **Cancel**.

Adding an HTTPS Port

The HTTPS port enables Integration Server to authenticate the client and server securely and encrypt the data exchanged. By default, the HTTPS listener uses the certificates for the default Integration Server SSL key. However, you can configure the listener to use its own private key residing in an Integration Server keystore (file- or SmartCard/HSM-based). For more information, see [“Configuring Server-Side SSL Settings” on page 463](#).

In addition, you can configure the type of client authentication that you want the server to perform. Client authentication allows you to verify the identity of the client (for more information, see [“Authenticating Clients” on page 515](#)).

➤ To add an HTTPS port

- Open Integration Server Administrator if it is not already open.
- In the **Security** menu of the Navigation panel, click **Ports**.
- Click **Add Port**.
- In the **Add Port** area of the screen, select **webMethods /HTTPS**.
- Click **Submit**. Integration Server Administrator displays a screen requesting information about the port.
- Under **Regular HTTPS Listener Configuration**, enter the following information:

For this parameter...	Specify...
Enable	Indicate whether to enable (Yes) or disable (No) this HTTPS or FTPS listener.
Port	The number you want to use for the port. Select a number that is not already in use on this host machine. Important: If you are running multiple Integration Servers on the same host machine, make sure the port numbers used on each server are unique.
Alias	An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a-z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).

For this parameter...	Specify...
Description	A description of the port.
Package name	<p>Package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port.</p> <p>If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.</p>
Bind Address (optional)	IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.
Backlog	<p>The number of requests that can remain in the queue for an enabled port before Integration Server begins rejecting requests. The default is 200. The maximum value is 65535.</p> <p>Note: This parameter does not apply to disabled ports. Integration Server refuses requests sent to disabled ports.</p>
Keep Alive Timeout	When to close the connection if the server has not received a request from the client within this timeout value (in milliseconds); or when to close the connection if the client has explicitly placed a close request with the server.
Threadpool	<p>Whether the listener will use this pool exclusively for dispatching requests. The existing Integration Server thread pool is a global thread pool. If there is a very high load on this resource, the user may have to wait for the global thread pool to process his request. However, with the private thread pool option enabled, requests coming into this port will not have to compete with other server functions for threads.</p> <p>Click Enable to enable the private thread pool settings. You can change or accept the default settings given below:</p> <p>Threadpool Min refers to the minimum number of threads for this private threadpool. The default is 1.</p> <p>Threadpool Max refers to the maximum number of threads for this private thread pool. The default is 5.</p>

For this parameter...	Specify...
	<p>Threadpool Priority refers to the Java thread priority. The default is 5.</p> <p>Important: Use this setting with extreme care because it will affect server performance and throughput.</p> <p>If you do not need to use the Threadpool feature, click Disable.</p> <p>When you view the port's details, the server reports the total number of private threadpool threads currently in use for the port.</p>

7. Under **Security Configuration**, enter the following information:

For this parameter...	Specify...
Use JSSE	<p>If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure Socket Extension (JSSE) socket factory. The default is Yes.</p> <p>If you set this value to No, the port supports only SSL 3.0 and TLS 1.0.</p> <p>Note: To control the cipher suites used on Integration Server ports that use JSSE and handle inbound requests, set the <code>watt.net.jsse.server.enabledCipherSuiteList</code>. For more information, see “Server Configuration Parameters” on page 1017.</p>

Client Authentication	Option	Description
	Username/Password	Integration Server prompts the client for a user ID and password.
	Digest	Integration Server uses password digest for authentication of all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the

For this parameter...	Specify...
Request Client Certificates	<p>client provides the required authentication information, Integration Server verifies and validates the request.</p>
	<p>Note: A port that is configured to use password digest for authentication of client requests will process a request from a user only if the user is configured to allow password digest for authentication. For more information about configuring a user for digest authentication, see “Adding User Accounts” on page 86.</p>
Require Client Certificates	<p>Integration Server requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. If the client provides a certificate:</p> <ul style="list-style-type: none"> ■ The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in Integration Server. If not, the client request fails, unless central user management is configured. ■ If central user management is configured, the server checks whether the certificate is mapped to a user in the central user database. If so, the server logs the client on as that user. If not, the client request fails. <p>Integration Server requires client certificates for all requests. The server behaves as described for Request Client Certificates, except that the</p>

For this parameter...**Specify...**

client must always provide a certificate.

Use Identity Provider

Integration Server uses an OpenID Provider to authenticate requests. Integration Server redirects all requests sent to this port to the OpenID Provider specified in **Identity Provider**.

Request Kerberos Ticket

Integration Server looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server tries to authenticate using the client certificates from the SSL handshake. If there are no client certificates, Integration Server, uses user name and password for basic authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.

Require Kerberos Ticket

Integration Server looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server fails the authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.

For this parameter...	Specify...
Kerberos Properties (Optional)	<p>Kerberos properties are used to enable Kerberos authentication by providing Kerberos-related details that will be used for handling service requests that come with a Kerberos ticket. For information on configuring Kerberos authentication, see “Kerberos Authentication” on page 517.</p> <p>JAAS Context Specify the custom JAAS context used for Kerberos authentication.</p> <p>In the following example, JAAS Context is <code>KerberosClient</code>:</p> <pre data-bbox="974 630 1461 955"> KerberosClient { com.sun.security.auth.module. Krb5LoginModule required useKeyTab=true keyTab=alice.keytab; }; </pre> <p>The <code>is_jaas.cnf</code> file distributed with Integration Server includes a JAAS context named <code>IS_KERBEROS_INBOUND</code> that can be used with inbound requests.</p>
Principal	<p>Specify the name of the principal to use for Kerberos authentication.</p>
Principal Password	<p>Specify the password for the principal that is used to authenticate the principal to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for authorization. The passwords may be encrypted using different encryption algorithms.</p> <p>If the JAAS login context contains <code>useKeyTab=false</code>, you must specify the principal password.</p>
Retype Principal Password	<p>Re-enter the principal password.</p>

For this parameter...	Specify...
	<p>Service Principal Name Format Displays username, which indicates that the principal name of the service is represented as a named user defined in the LDAP or central user directory used for authentication to the KDC.</p> <p>Service Principal Name Specify the name of the principal used with the service that the Kerberos client wants to access. Specify the Service Principal Name in the following format:</p> <p style="text-align: center;"><i>principal-name.instance-name@realm-name</i></p>

8. Under **Listener Specific Credentials**, enter the following information:

Note:

Use these settings only if you want to use a different set of credentials from the ones specified on the Certificates Screen.

For this parameter...	Specify...
Keystore Alias	<p>Optional. A user-specified, text identifier for an Integration Server keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p> <p>For more information, see “Creating Keystore Aliases” on page 481.</p>
Key Alias	<p>Optional. The alias for the private key, which must be stored in the keystore specified by the above keystore alias.</p>
Truststore Alias	<p>Optional. The alias for the truststore. The truststore must contain the trusted root certificate for the CA that signed Integration Server certificate associated with the key alias. The truststore also contains the list of CA certificates that Integration Server uses to validate the trust relationship.</p>

9. Click **Save Changes**.
10. On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

11. On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

About File Polling Ports

A file polling port periodically polls a monitoring directory for the arrival of files. Integration Server then runs a special file processing service against the file. A file polling port and processing works in the following way:

1. A file polling port on Integration Server periodically polls a monitoring directory for the arrival of files. Each file polling port polls only one monitoring directory.
2. When it detects a new file, Integration Server moves the file to a working directory. Integration Server renames the file to include a random internal counter which ensures file name uniqueness. The resulting filename in the working directory adheres to the following naming convention: FilePolling.<counter>.<file_Name>

Note:

All files in the working, completion, and error directory follow the above naming convention.

3. Integration Server executes the file processing service specified for the port. The service might parse, convert, and validate the file then write it to the file system. This service, which you write, is the only service that can be invoked through this port.
4. One of the following happens depending on whether or not the service processes the file successfully:
 - If processing of file completes successfully, Integration Server moves the file to the completion directory.
 - If processing of the file ends with an error, Integration Server moves the file to an error directory.
5. Integration Server cleans up the completion and error directories at a regular interval. You can configure the how old a file must be before it can be removed from the completion or error directories and how frequently Integration Server cleans up the completion and error directories.

Note:

If Integration Server shuts down or becomes unavailable before completing processing of a file in the working directory, the file remains in the working directory after restart. Integration Server will not restart or resume processing of a file left in the working directory. The administrator needs to decide whether or not to move the file from the working directory to the monitoring directory or to delete the file.

In a cluster of Integration Servers or non-clustered group of Integration Servers, file polling works much the same way as it does on an individual Integration Server. The only difference is that more than one Integration Server polls the monitoring directory. Once an Integration Server in a group

retrieves a file from the monitoring directory, the file is not available to other Integration Servers in the group.

To configure file polling, you must do the following:

- Set up the Monitoring Directory on Integration Server. Other directories used for file polling are automatically created by Integration Server.
- Write a file processing service and make it available to Integration Server. See *webMethods Service Development Help* and the *Flat File Schema Developer's Guide* for examples of such services.
- Set up the file polling port on Integration Server.

Adding a File Polling Port

Use the following procedure to add a file polling port to Integration Server.

> To add a file polling port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Add Port**.
4. In the **Add Port** area of the screen, select **webMethods /FilePolling**.
5. Click **Submit**. Integration Server Administrator displays a screen requesting information about the port.
6. Under **Package**, enter the following information:

For this parameter...	Specify...
-----------------------	------------

Package Name	The package associated with this port.
---------------------	--

When you enable the package, the server enables the port. When you disable the package, the server disables the port. If you are performing special file handling, specify the package that contains the services that perform that processing. If you want to process flat files from this port, select **WmFlatFile**, which contains built-in services you can use to process flat files.

Note:

If you replicate this package, whether to a server on the same machine or a server on a separate machine, a file polling port with the same settings is created on the target

For this parameter... Specify...

server. If a file polling port already exists on the target server, its settings remain intact. If the original and target servers reside on the same machine, they will share the same monitoring directory. If the target server resides on another machine, by default, another monitoring directory will be created on the target server's machine.

Alias

An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).

Description

A description of the port.

7. Under **Polling Information**, enter the following information:

For this parameter...	Specify...
Monitoring Directory	Directory on Integration Server that you want to monitor for files.
Working Directory (optional)	Directory on Integration Server to which the server should move files for processing after they have been identified in the Monitoring Directory . Files must meet age and file name requirements before being moved to the Working Directory . The default sub-directory, <i>MonitoringDirectory \ . . \Work</i> , is automatically created if no directory is specified.
Completion Directory (optional)	Directory on Integration Server to which you want files moved when processing is completed in the Monitoring Directory or Working Directory . The default sub-directory, <i>MonitoringDirectory \ . . \Done</i> , is automatically created if no directory is specified.

Note:

You can instruct Integration Server to append a time stamp to a file before moving it to the **Completion Directory**. To do this, update the properties.cnf file to include `appendTimeStampToFileName=true`. The resulting name follows this convention:

```
FilePolling.<timestamp>.<counter>.<file_Name>
```

The properties.cnf file is located here:

```
Integration Server_directory
\instances\instanceName\packages\WmFlatFile\config
```

If the properties.cnf file does not exist, create it.

For this parameter...	Specify...
Error Directory (optional)	Directory on Integration Server to which you want files moved when processing fails. The default sub-directory, <i>MonitoringDirectory \ . . \Error</i> , is automatically created if no directory is specified.
File Name Filter (optional)	The file name filter for files in the Monitoring Directory . Integration Server only processes files that meet the filter requirements. If you do not specify this field, all files will be polled. You can specify pattern matching in this field.
File Age (optional)	The minimum age (in seconds) at which a file in the Monitoring Directory can be processed. Integration Server determines file age based on when the file was last modified on the monitoring directory. You can adjust this age as needed to make sure the server does not process a file before the entire file has been copied to the Monitoring Directory. The default is 0.
Content Type	<p>Content type to use for the file. The server uses the content handler associated with the content type specified in this field. If no value is specified, Integration Server performs MIME mapping based on the file extension.</p> <p>The File Polling port handles XML content in the following manner:</p> <ul style="list-style-type: none"> ■ If the Content Type is 'text/xml', the XML content is passed as 'node' {com.wm.lang.xml.Document} to the pipeline. ■ If the Content Type is not set, the default MIME mapping of 'text/xml' is used. In this case also, the XML content is passed as 'node' {com.wm.lang.xml.Document} to the pipeline. ■ If the Content Type is 'application/xml', the XML content is passed as 'ffdata' {java.io.BufferedInputStream} to the pipeline.

Note:


To override the default encoding, update the properties.cnf file to include `filepollingport=encodingType` where *encodingType* is the encoding that you want the flat file parser to use for parsing XML files.

The properties.cnf file is located here:

```
Integration Server_directory
\instances\instanceName\packages\WmFlatFile\config
```

If the properties.cnf file does not exist, create it.

For this parameter...	Specify...
Allow Recursive Polling	Whether Integration Server is to poll all sub-directories in the Monitoring Directory . Select Yes or No .
Enable Clustering	Whether Integration Server should allow clustering in the Monitoring Directory. Select Yes or No .
Lock File Extension (optional)	Defines the polling for a particular extension. The default is lock.
Number of files to process per interval (optional)	Specifies the maximum number of files that the file polling listener can process per interval. When you specify a positive integer, the file polling listener processes only that number of files from the monitoring directory. Any files that remain in the monitoring directory will be processed during subsequent intervals. If you specify -1 or do not specify a value, the listener processes all of the files in the monitoring directory. The default is -1.

8. Under **Security**, in the **Run services as user** parameter, specify the user name you want to use to run the services assigned to the file polling directory. Click  to lookup and select a user. The user can be an internal or external user.
9. Under **Message Processing**, supply the following information:

For this parameter...	Specify...
Enable	Whether to enable (Yes) or disable (No) this file polling port.
Processing Service	Name of the service you want Integration Server to execute for polled files. The server executes this service when the file has been copied to the Working directory. This service should be the only service available from this port.
	<p>Important: If you change the processing service for a file polling port, you must also change the list of services available from this port to contain just the new service. See below for more information.</p>
File Polling Interval	How often (in seconds) you want Integration Server to poll the Monitoring Directory for files.
Log Only When Directory Availability Changes	<p>If you select No (the default), the listener will log a message every time the monitoring directory is unavailable.</p> <p>If you select Yes, the listener will log a message in either of the following cases:</p>

For this parameter...	Specify...
	<ul style="list-style-type: none">■ The directory <i>was</i> available during the last polling attempt but <i>not</i> available during the current attempt■ The directory was <i>not</i> available during the last polling attempt but <i>is</i> available during the current attempt
Directories are an NFS Mounted File System	<p>For use on a UNIX system where the monitoring directory, working directory, completion directory, and/or error directory are network drives mounted on the local file system.</p> <p>If you select No (the default), the listener will call the Java File.renameTo() method to move the files from the monitoring directory to the working directory, and from the working directory to the completion and/or error directory.</p> <p>If you select Yes, the listener will first call the Java File.renameTo() method to move the files from the monitoring directory. If this method fails, the listener will then copy the files from the monitoring directory to the working directory and delete the files from the monitoring directory. This operation will fail if either the copy action or the delete action fails. The same behavior applies when moving files from the working directory to the completion and/or error directory.</p>
Cleanup Service (Optional)	The name of the service that you want to use to clean up the directories specified under Polling Information .
Cleanup At Startup	Whether to clean up files that are located in the Completion Directory and Error Directory when the file polling port is started.
Cleanup File Age (Optional)	The number of days to wait before deleting processed files from your directories. The default is 7 days.
Cleanup Interval (Optional)	How often (in hours) you want Integration Server to check the processed files for cleanup. The default is 24 hours
Maximum Number of Invocation Threads	The number of threads you want Integration Server to use for a port. You can specify a value between 1 and 10, inclusive.

10. Click **Save Changes**.

11. Make sure the port's access mode is properly set and that the file processing service is the only service accessible from the port.

- In the **Ports** screen, click **Edit** in the **Access Mode** field for the port you just created.
- Click **Set Access Mode to Deny by Default**.

- c. Click **Add Folders and Services to Allow List**.
- d. Type the name of the processing service for this port in the text box under **Enter one folder or service per line**.
- e. Remove any other services from the allow list.
- f. Click **Save Additions**.

Note:

If you change the processing service for a file polling port, remember to change the Allow List for the port as well. Follow the procedure described above to alter the allowed service list.

Adding an FTPS Port

The FTPS (FTP over SSL) port enables the server to authenticate the FTP client and server in a secure manner, and encrypt the control and data exchange between the FTP client and server.

Keep the following points in mind when configuring an FTPS port:

- FTPS clients are always prompted for a userid and password.
- By default, the FTPS port will work only with secure clients. A secure client is a client that secures the connection by issuing the AUTH command. You also can configure the FTPS listener to operate with clients that are not secure.
- You can configure the FTPS port to use its own certificate or use Integration Server certificate, or to request or require client certificates. In addition, you can configure the listener to use a private key and certificate chain residing in a keystore (file- or SmartCard/HSM-based). For more information about client certificates, see [“Authenticating Clients” on page 515](#).
- By default, Integration Server does not perform certificate mapping for FTPS ports. To use this feature, you must set the `watt.net.ftpUseCertMap` configuration property to true. For more information about how client authentication works for FTPS ports, see [“Authenticating Clients” on page 515](#). For more information about certificate mapping, see [“Importing a Certificate \(Client or CA Signing Certificate\) and Mapping It to a User” on page 519](#).
- When a user logs in through an FTPS port, Integration Server can place the user in the default FTP root directory or in the client user directory. Integration Server chooses the directory based on the setting of the `watt.server.login.userFtpDir` parameter. For more information, see [“Server Configuration Parameters” on page 1017](#).

➤ To add an FTPS port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.

3. Click **Add Port**.
4. In the **Add Port** area of the screen, select **webMethods /FTPS**.
5. Click **Submit**. Integration Server displays a screen requesting information about the port. Enter the following information:

For this parameter...	Specify...
Enable	Select whether to enable (Yes) or disable (No) this FTPS port.
Port	The number you want to use for the port. Select a number that is not already in use on this host machine. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: If you are running multiple Integration Servers on the same host machine, make sure the port numbers used on each server are unique.</p> </div>
Alias	An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.
Package Name	Package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port. If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.
Bind Address (optional)	IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.
Passive Mode Listen Address (optional)	Address to be sent by the PORT command. You can specify a host name or IP address. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This option is not applicable when the FTPS port is bound to an IPv6 address. In that case, the passive mode listen address is the same as the port bind address.</p> </div>

For this parameter...**Specify...**

When running in passive mode, the FTPS port sends a PORT command to the FTPS client. The PORT command specifies the address and port to which the client should connect to create a data connection. If the FTPS port is behind a NAT server, however, the address of the host on which Integration Server runs is not visible to the FTPS client. Consequently the PORT command does not contain the information the client needs to connect to the server. To remedy this situation, you can specify a value for the `watt.net.ftpPassiveLocalAddr` property in the server configuration file (`server.cnf`), which is located in the *Integration Server_directory* \instances\instance_name\config directory (see [“Server Configuration Parameters” on page 1017](#)).

Alternatively, you can use the **Passive Mode Listen Address** field to specify the passive mode address for an individual FTPS port. That way, you can specify a different passive mode address for each FTPS port. If an address is specified in the **Passive Mode Listen Address** field and in the `watt.net.ftpPassiveLocalAddr` property, the PORT command uses the value specified in the `watt.net.ftpPassiveLocalAddr` property.

Secure Clients Only

Select this check box to prevent the FTPS listener from operating with non-secure clients.

6. Under **Security Configuration**, enter the following information:

For this parameter...**Specify...****Use JSSE**

If this port should support TLS 1.1 or TLS 1.2, click **Yes** to create the port using the Java Secure Socket Extension (JSSE) library. The default is **Yes**.

If you set this value to **No**, the port supports only SSL 3.0 and TLS 1.0 and Entrust IAIK library is used to create the outbound FTPS connection.

Note:

To control the cipher suites used on Integration Server ports that use JSSE and handle inbound requests, set the `watt.net.jsse.server.enabledCipherSuiteList`. For more information, see [“Server Configuration Parameters” on page 1017](#).

For this parameter...	Specify...								
Client Authentication	The type of client authentication you want Integration Server to perform for requests that arrive on this FTPS port. Select one of the following:								
	<table border="1"> <thead> <tr> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Username/Password</td> <td>Integration Server prompts the client for a user ID and password.</td> </tr> <tr> <td>Request Client Certificates</td> <td> <p>Integration Server requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. If the client provides a certificate:</p> <ul style="list-style-type: none"> ■ The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in Integration Server. If not, the client request fails, unless central user management is configured. ■ If central user management is configured, the server checks whether the certificate is mapped to a user in the central user database. If so, the server logs the client on as that user. If not, the client request fails. </td> </tr> <tr> <td>Require Client Certificates</td> <td>Integration Server requires client certificates for all requests. The server behaves as described for Request Client Certificates, except that the client must always provide a certificate.</td> </tr> </tbody> </table>	Option	Description	Username/Password	Integration Server prompts the client for a user ID and password.	Request Client Certificates	<p>Integration Server requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. If the client provides a certificate:</p> <ul style="list-style-type: none"> ■ The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in Integration Server. If not, the client request fails, unless central user management is configured. ■ If central user management is configured, the server checks whether the certificate is mapped to a user in the central user database. If so, the server logs the client on as that user. If not, the client request fails. 	Require Client Certificates	Integration Server requires client certificates for all requests. The server behaves as described for Request Client Certificates , except that the client must always provide a certificate.
Option	Description								
Username/Password	Integration Server prompts the client for a user ID and password.								
Request Client Certificates	<p>Integration Server requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. If the client provides a certificate:</p> <ul style="list-style-type: none"> ■ The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in Integration Server. If not, the client request fails, unless central user management is configured. ■ If central user management is configured, the server checks whether the certificate is mapped to a user in the central user database. If so, the server logs the client on as that user. If not, the client request fails. 								
Require Client Certificates	Integration Server requires client certificates for all requests. The server behaves as described for Request Client Certificates , except that the client must always provide a certificate.								

7. Under **Listener Specific Credentials**, enter the following information:

Note:

Use these settings only if you want to use a different set of credentials from the ones specified on the Certificates Screen.

For this parameter...	Specify...
Keystore Alias	<p>Optional. A user-specified, text identifier for an Integration Server keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p> <p>For more information, see “Creating Keystore Aliases” on page 481.</p>
Key Alias	<p>Optional. The alias for the private key, which must be stored in the keystore specified by the above keystore alias.</p>
Truststore Alias	<p>Optional. The alias for the truststore. The truststore must contain the trusted root certificate for the CA that signed Integration Server certificate associated with the key alias. The truststore also contains the list of CA certificates that Integration Server uses to validate the trust relationship.</p>

- Click **Save Changes**.
- On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

- On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Adding an FTP Port

Using an FTP port, you can move files to and from Integration Server. Complete the instructions below to configure an FTP port.

When a user logs in through an FTP port, Integration Server can place the user in the default FTP root directory or in the client user directory. Integration Server chooses the directory based on the setting of the `watt.server.login.userFtpDir` parameter. For more information, see [“Server Configuration Parameters” on page 1017](#).

➤ To add an FTP port

- Open Integration Server Administrator if it is not already open.

2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Add Port**.
4. In the **Add Port** area of the screen, select **webMethods /FTP**.
5. Click **Submit**. Integration Server displays a screen requesting information about the port. Enter the following information:

For this parameter...	Specify...
Port	<p>The number you want to use for the port. Select a number that is not already in use on this host machine.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: If you are running multiple Integration Servers on the same host machine, make sure the port numbers used on each server are unique.</p> </div>
Alias	<p>An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p>
Description	<p>A description of the port.</p>
Package Name	<p>Package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port.</p> <p>If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.</p>
Bind Address (optional)	<p>IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.</p>
Passive Mode Listen Address (optional)	<p>The address that should be sent by the PORT command. A host name or IP address can be specified.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This option is not applicable when the FTP port is bound to an IPv6 address. In that case, the passive mode listen address is the same as the port bind address.</p> </div>

For this parameter...**Specify...**

When running in passive mode, the FTP port sends a PORT command to the FTP client. The PORT command specifies the address and port to which the client should connect to create a data connection. If the FTP port is behind a NAT server, however, the address of the host on which Integration Server runs is not visible to the FTP client. Consequently the PORT command does not contain the information the client needs to connect to the server. To remedy this situation, you can specify a value for the `watt.net.ftpPassiveLocalAddr` property in the server configuration file (`server.cnf`), which is located in the *Integration Server_directory \instances\instance_name\config* directory (see [“Server Configuration Parameters”](#) on page 1017).

Alternatively, you can use the **Passive Mode Listen Address** field to specify the passive mode address for an individual FTP port. That way, you can specify a different passive mode address for each FTP port. If an address is specified in the **Passive Mode Listen Address** field and in the `watt.net.ftpPassiveLocalAddr` property, the PORT command uses the value specified in the `watt.net.ftpPassiveLocalAddr` property.

6. Click **Save Changes**.
7. On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port”](#) on page 487

8. On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Adding an E-Mail Port

By setting up one or more e-mail ports on your Integration Server, you can receive client requests through an e-mail server (POP3 or IMAP). The client builds an e-mail that contains the name of the service to run and parameters to pass to the service. The e-mail can also contain user ID and password information.

Before adding an e-mail port, review the information in [“Security Considerations for E-Mail Ports”](#) on page 189.

> To add an e-mail port

1. Open Integration Server Administrator if it is not already open.

- In the **Security** menu of the Navigation panel, click **Ports**.
- Click **Add Port**.
- In the **Add Port** area of the screen, select **webMethods /E-mail**.
- Click **Submit**. Integration Server displays the **Edit E-mail Client Configuration** screen requesting information about the port.
- Under **Package**, specify the following information:

For this parameter...	Specify...
Package Name	<p>The package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port.</p> <p>If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.</p>
Alias	<p>An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p>
Description	<p>A description of the port.</p>

- Under **Server Information**, supply the following information:

For this parameter...	Specify...
Enable	Whether to enable (Yes) or disable (No) the e-mail port. The port can receive requests only when it is enabled.
Type	Type of mail server. Select POP3 or IMAP .
Host Name	Name of the machine on which the POP3 or IMAP server is running.
Port	Port on the e-mail server to which Integration Server is to connect. For a POP3 mail server, the defaults are 110 for explicit SSL and 995 for implicit SSL.

For this parameter...	Specify...
	For an IMAP mail server, the defaults are 143 for explicit SSL and 993 for implicit SSL.
User Name	User name that identifies you to the e-mail server.
Authentication	<p>The type of authentication that Integration Server should use to connect to the specified e-mail server. Select Basic Authentication or OAuth.</p> <ul style="list-style-type: none"> ■ Select Basic Authentication if you want to authenticate the user on the specified e-mail server with only login credentials (username and password). ■ Select OAuth to authenticate the user on the specified e-mail server using the credentials issued by the OAuth server. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Only the Authorization Code grant type is supported for OAuth.</p> <p>Note: To obtain the OAuth credentials for Integration Server, you must first register Integration Server with the OAuth Server. For details about registering an application and obtaining the OAuth credentials, see your authorization server's documentation. If you are using Microsoft Azure Active Directory as your OAuth server, see this article.</p> </div>
Password	The password associated with the username that identifies you on the e-mail server. This field applies only to Basic Authentication.
Auth URL	The URL of the endpoint that Integration Server must use to request authorization code. This field applies only to OAuth.
Client ID	The unique public identifier that the OAuth server generates for Integration Server during registration. This field applies only to OAuth.
Client Secret	The unique string that the OAuth server provides to Integration Server during registration. It is only known to Integration Server and the OAuth server. This field applies only to OAuth.
Scope	The e-mail server access permissions configured for Integration Server during registration. You can specify multiple scopes separated by a space. This field applies only to OAuth.

For this parameter...	Specify...						
Access Token URL	The URL of the endpoint that Integration Server must use to request an access token from the OAuth server. This field applies only to OAuth.						
Redirect URL	<p>The URL that the OAuth server must use to send authentication responses to Integration Server.</p> <ul style="list-style-type: none"> ■ If you are accessing Integration Server locally, enter the redirect URLs in one of the applicable formats: <pre>http://localhost:{port}/WmRoot/security-oauth-get-authcode.dsp</pre> or <pre>https://localhost:{port}/WmRoot/security-oauth-get-authcode.dsp</pre> ■ If you are accessing Integration Server remotely, use <pre>https://{ISHostName}:{port}/WmRoot/security-oauth-get-authcode.dsp</pre> <p>This field applies only to OAuth.</p>						
Authorization Code	<p>Click Get Authorization Code to receive a unique string from the OAuth Server that Integration Server requires to request an access token.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Whenever you edit the OAuth credentials for an email port, ensure to get the authorization code.</p> </div>						
Access Token Expiry Time	Specifies the time until when an access token issued to webMethods Integration Server is valid. This field applies only to OAuth. This is a read-only field and is generated based on the access token.						
Transport Layer Security	Type of SSL encryption that Integration Server uses when communicating with an e-mail server. You can configure the port to use explicit, implicit, or no Transport Layer Security.						
	<table border="1"> <thead> <tr> <th>Specify...</th> <th>To...</th> </tr> </thead> <tbody> <tr> <td>None</td> <td> <p>Default. Use a non-secure mode when communicating with an e-mail server.</p> <p>When you specify None, the e-mail server authenticates the e-mail client using only the username and password.</p> </td> </tr> <tr> <td>Explicit</td> <td>Use explicit security when communicating with an e-mail server. With explicit security, Integration Server establishes an un-encrypted connection to the e-mail server and then upgrades to the secure mode.</td> </tr> </tbody> </table>	Specify...	To...	None	<p>Default. Use a non-secure mode when communicating with an e-mail server.</p> <p>When you specify None, the e-mail server authenticates the e-mail client using only the username and password.</p>	Explicit	Use explicit security when communicating with an e-mail server. With explicit security, Integration Server establishes an un-encrypted connection to the e-mail server and then upgrades to the secure mode.
Specify...	To...						
None	<p>Default. Use a non-secure mode when communicating with an e-mail server.</p> <p>When you specify None, the e-mail server authenticates the e-mail client using only the username and password.</p>						
Explicit	Use explicit security when communicating with an e-mail server. With explicit security, Integration Server establishes an un-encrypted connection to the e-mail server and then upgrades to the secure mode.						



For this parameter...	Specify...
	<p>With explicit Transport layer Security, Integration Server can communicate with e-mail servers that support and do not support SSL encryption. If the e-mail server does not support Transport Layer Security, Integration Server will disconnect the connection established to the e-mail server. You can then establish an un-secure connection with the e-mail server by selecting the None option in the Transport Layer Security field and enabling the port.</p>
	<p>Implicit Use implicit security when communicating with an e-mail server. With implicit security, Integration Server always establishes an encrypted connection to the e-mail server. Only clients that support SSL will be permitted access.</p>
Truststore Alias (optional)	<p>Optional. Alias for the truststore that contains certificates presented by the e-mail server to Integration Server. If you do not select a truststore alias, the default truststore alias specified in the <code>watt.security.trustStoreAlias</code> property will be used. For more information about this property, see “watt.security.” on page 1048. For more information about truststore alias, see “Creating Truststore Aliases” on page 482.</p>
Time Interval	<p>How often (in seconds) Integration Server is to check for e-mails in the POP3 or IMAP server.</p>
Log out after each mail check	<p>For use with IMAP and multithreading only. If you select Yes, Integration Server logs out a read-only thread to the IMAP mail server after checking for mail on that thread. The main read/write thread to the IMAP server remains intact. If you select No, all the read-only threads remain intact. Select Yes if your IMAP server restricts the number of connections it will allow to remain logged in.</p>

Note:

OAuth-based authentication for email ports was introduced for PIE-66302 in IS_10.5_Core_Fix7.

8. Under **Security**, provide the following information:

For this parameter...	Specify...
Run services as user	<p>If you select Yes in the Require authentication within message field, the Run services as user field remains blank because Integration Server expects the user name and password to be in</p>

For this parameter...	Specify...
	the e-mail. If you select No in the Require authentication within message field, you must enter the user under which the service is to run on Integration Server.
Require authentication within message	<p>If you select Yes, Integration Server checks for \$user and \$pass parameters in the Subject line of the e-mail. The user name is the user under which the service is to run on Integration Server. If you select No, you must specify the user in the Run services as user field above.</p> <p>When you select No,  appears next to this field. Click  to look up and select a user. The user can be an internal or external user.</p>
Use JSSE	<p>Select Yes to use the JSSE library to create a port that supports TLS 1.1 or TLS 1.2.</p> <p>Select No to use the Entrust IAIK library to create a port that supports only SSL 3.0 and TLS 1.0.</p> <p>Note: JSSE library support introduced for PIE-61569 in IS_10.5_Core_Fix4.</p>

9. Under **Message Processing**, complete the following information:

For this parameter...	Specify...
Global Service (optional)	Service to be executed on Integration Server. This field overrides a service specified in the Subject line of the e-mail.
Default Service (optional)	Service to be executed if the e-mail does not provide a valid service in the Subject line <i>and</i> the Global Service field is blank.
Send reply e-mail with service output	Click Yes if you want Integration Server to send any output generated by the service to the original sender in an e-mail attachment. Click No if you do not want to do so. If the original e-mail contained multiple attachments, the reply contains an equal number of attachments.
Send reply e-mail on error	Click Yes if you want Integration Server to report any errors that occurred during service execution to the original sender in the Body portion of an e-mail. Click No if you do not want to do so.
	<p>Note: For more information about sending e-mail notifications when errors occur, see “Sending Messages About Critical Issues to E-mail Addresses” on page 220.</p>

For this parameter...	Specify...
Delete valid messages (IMAP only)	Click Yes if you want to delete a valid e-mail from the IMAP server once Integration Server has successfully received the e-mail. This setting helps prevent e-mails from accumulating on the IMAP server, possibly affecting disk space and performance. Integration Server always deletes e-mails on a POP3 server. Click No if you want to retain the e-mails on the IMAP server.
Delete invalid messages (IMAP only)	<p>Click Yes if you want to delete invalid e-mails from the IMAP server. Click No if you do want to remove these e-mails from the server.</p> <p>Invalid e-mails are those that reference services that cannot be invoked. For example, if the referenced service does not exist, the server will delete the e-mail. If the service was invoked, but encountered errors, the server considers the associated e-mail to be valid.</p> <p>This setting helps prevent invalid e-mails from accumulating on the IMAP server, possibly affecting disk space and performance. Integration Server always deletes e-mails on a POP3 server.</p>
Multithreaded processing (IMAP only)	Click Yes if you want Integration Server to use multiple threads for this port. This setting allows the port to handle multiple requests at once and avoid a bottleneck. Click No if you do not need this feature.
Number of threads if multithreading is turned on.	Tells Integration Server the number of threads to use for this port. The default is set to 0.
	<p>Note:</p> <p>If the e-mail port is configured to allow multithreading and if the e-mail port receives a large number of messages in a short period of time, the e-mail port will monopolize the server thread pool for retrieving and processing messages. This will slow down the performance of Integration Server. To avoid this, you can limit the number of threads used to process messages received by the e-mail port by setting the <code>watt.server.email.waitForServiceCompletion</code> property. For more information about this property, see “Server Configuration Parameters” on page 1017.</p>
Invoke service for each part of multipart message	Specifies whether Integration Server invokes the service for each part of a multi-part message or just once for the entire message.

For this parameter...	Specify...
	<p>If you specify No, the entire e-mail is passed to the appropriate content handler and then to the specified service for execution. When you send an entire multi-part e-mail, make sure the server includes the e-mail headers from the beginning of the message, so that the content handler and/or service knows how to process the content type headers included in each part of the e-mail. See Include e-mail headers when passing message to content handler below.</p> <p>If you specify Yes, Integration Server treats each part of the message individually. That is, Integration Server sends each part to the content handler and then to the specified service. When you specify Yes, you probably do not want to include the e-mail headers from the beginning of the message, because each section has its own headers that the content handler and/or the service already knows how to process. See Include e-mail headers when passing message to content handler below.</p>
Include e-mail headers when passing message to content handler	<p>Specifies whether Integration Server includes the e-mail headers when passing an e-mail message to the content handler. The e-mail headers are typically found at the beginning of an e-mail message. Specify Yes if you are processing a multi-part message as a single message. This ensures that the content handler and/or service can properly process the body of the e-mail. Specify No if you are processing the different parts of an e-mail individually. If you are processing a single-part e-mail, you probably do not want to include e-mail headers.</p>
E-mail body contains URL encoded input parameters	<p>Specifies how Integration Server treats input parameters it finds in e-mail messages. With this value set to Yes, Integration Server considers a string such as <code>?one=1+two=2</code> to be a URL encoded input parameter. It then decodes this string into an IData object, puts it into the pipeline, and passes it to the service. With this value set to No, Integration Server treats the string as plain text and passes it to the appropriate content handler.</p>

Note:

Once Integration Server successfully processes an email message, the email message is not re-processed if moved across folders.

- Click **Save Changes**. If you have configured OAuth, Integration Server receives the access token and displays the **Access Token Expiry Time**.

Note:

Whenever you create or edit an email port, you must clear the browser cache. This applies only if you are using the OAuth authentication type.

11. On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

Note:

If you set port access restrictions, be sure the `watt.net.email.validateHost` server configuration property is set to `true`, so Integration Server honors your IP access restrictions.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

12. On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Security Considerations for E-Mail Ports

Passing a userid and password in an e-mail presents a possible security exposure. While the e-mail resides on the POP3 or IMAP server, someone might be able to access the information in the e-mail using the userid and password. To make your communication with the e-mail server secure, you can configure Integration Server to use either explicit or implicit Transport Layer Security (TLS) through Secure Sockets Layer (SSL). Integration Server uses client certificates for implementing Transport Layer Security and authenticates the connection using the userid and password.

Adding an HTTP Diagnostic Port

The diagnostic port is a special port that uses threads from a dedicated thread pool to accept requests via HTTP. The diagnostic port uses a dedicated thread pool so that you can access Integration Server when it becomes unresponsive.

When you install Integration Server, it automatically creates the diagnostic port at 9999. If another port is running on that Integration Server at 9999, the server will disable the diagnostic port at startup.

Each Integration Server can have only one diagnostic port. If you want to add a new diagnostic port, you must delete the existing port first. For information about how to delete a port, see [“Deleting a Port” on page 205](#).

If you are running multiple Integration Servers on the same machine, you specified the diagnostic port number for each server instance during the instance creation process. If the diagnostic port numbers are not unique between Integration Server instances, the first Integration Server to start on the machine will have a functioning diagnostic port, but Integration Servers that start after the first one will not. For more information about running multiple Integration Servers on the same machine, see [“Running Multiple Integration Server Instances” on page 63](#).

For more information about the diagnostic port, see [“Diagnosing the Integration Server ” on page 937](#).

➤ **To add an HTTP diagnostic port**

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Add Port**.
4. Under **Add Port**, select **HTTP Diagnostic**.
5. Click **Submit**.
6. On the **Edit Diagnostic Port Configuration** screen, enter the following information:

For this parameter	Specify
Port	<p>The number you want to use for the diagnostic port. Select a number that is not already in use on this host machine.</p> <p>Note: The <code>watt.server.diagnostic.port</code> server configuration parameter overrides this port number.</p> <p>Important: If you are running multiple Integration Servers on the same host machine, make sure the diagnostic port number on each server is unique.</p>
Alias	<p>An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a-z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p>
Description	<p>A description of the port.</p>
Package Name	<p>The package associated with this port. The default package is <code>WmRoot</code>. When you enable the package, the server enables the port. When you disable the package, the server disables the port.</p> <p>If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.</p> <p>Note: You cannot change the Package Name associated with this port. The diagnostic port must always be associated with the <code>WmRoot</code> package.</p>

For this parameter	Specify
Bind Address (optional)	The IP address to which you want to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use a specific address. If you do not specify a bind address, the server picks one for you.
Backlog	The number of requests that can remain in the queue for an enabled port before Integration Server begins rejecting requests. The default is 200. The maximum value is 65535. Note: This parameter does not apply to disabled ports. Integration Server refuses requests sent to disabled ports.
Keep Alive Timeout	When to close the connection if the server has not received a request from the client within this timeout value (in milliseconds); or when to close the connection if the client has explicitly placed a close request with the server.
Threadpool	Whether the listener will use this pool exclusively for dispatching requests. The existing Integration Server thread pool is a global thread pool. If there is a very high load on this resource, the user may have to wait for the global thread pool to process his request. However, with the private thread pool option enabled, requests coming into this port will not have to compete with other server functions for threads. Click Enable to enable the private thread pool settings. You can change or accept the default settings given below: Threadpool Min refers to the minimum number of threads for this private threadpool. The default is 1. Threadpool Max refers to the maximum number of threads for this private threadpool. The default is 5. Threadpool Priority refers to the Java thread priority. The default is 5. Important: Use this setting with extreme care because it will affect server performance and throughput. If you do not need to use the Threadpool feature, click Disable . When you view the port's details, the server reports the total number of private threadpool threads currently in use for the port.

- Under **Security Configuration**, enter the following information:

For this parameter...	Specify...
Client Authentication	<p>The type of client authentication you want Integration Server to perform for requests that arrive on this HTTP port. Select one of the following:</p>
Option	Description
Username/Password	<p>Integration Server prompts the client for a user ID and password.</p>
Digest	<p>Integration Server uses password digest to authenticate all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p> <div data-bbox="865 947 1360 1318" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: A port that is configured to use password digest for authentication of client requests will process a request from a user only if the user is configured to allow password digest for authentication. For more information about configuring a user for digest authentication, see “Adding User Accounts” on page 86.</p> </div>
Request Kerberos Ticket	<p>Integration Server looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server uses user name and password for basic authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p>

For this parameter...	Specify...
	<p>Require Kerberos Ticket</p> <p>Integration Server looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server fails the authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p>
<p>Kerberos Properties(Optional)</p>	<p>Kerberos properties are used to enable Kerberos authentication by providing Kerberos-related details that will be used for handling service requests that come with a Kerberos ticket. For information on configuring Kerberos authentication, see “Kerberos Authentication” on page 517.</p>
	<p>JAAS Context</p> <p>Specify the custom JAAS context used for Kerberos authentication.</p> <p>In the following example, JAAS Context is KerberosClient:</p> <pre data-bbox="958 1134 1453 1470">KerberosClient { com.sun.security.auth.module. Krb5LoginModule required useKeyTab=true keyTab=alice.keytab; };</pre> <p>The is_jaas.cnf file distributed with Integration Server includes a JAAS context named IS_KERBEROS_INBOUND that can be used with inbound requests.</p>
	<p>Principal</p> <p>Specify the name of the principal to use for Kerberos authentication.</p>
	<p>Principal Password</p> <p>Specify the password for the principal that is used to authenticate the principal</p>

For this parameter...	Specify...
	to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for authorization. The passwords may be encrypted using different encryption algorithms.
	If the JAAS login context contains <code>useKeyTab=false</code> , you must specify the principal password.
Retype Principal Password	Re-enter the principal password.
Service Principal Name Format	Displays username , which indicates that the principal name of the service is represented as a named user defined in the LDAP or central user directory used for authentication to the KDC.
Service Principal Name	Specify the name of the principal used with the service that the Kerberos client wants to access. Specify the Service Principal Name in the following format: <i>principal-name.instance-name@realm-name</i>

- Click **Save Changes**.
- On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

- On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Adding an HTTPS Diagnostic Port

The diagnostic port is a special port that uses threads from a dedicated thread pool to accept requests via HTTP/S. The diagnostic port uses a dedicated thread pool so that you can access Integration Server when it becomes unresponsive.

Each Integration Server can have only one diagnostic port. If you want to add a new diagnostic port, you must delete the existing port first. For information about how to delete a port, see [“Deleting a Port” on page 205](#).

If you are running multiple Integration Servers on the same machine, you specified the diagnostic port number for each server instance during the instance creation process. If the diagnostic port numbers are not unique between Integration Server instances, the first Integration Server to start on the machine will have a functioning diagnostic port, but Integration Servers that start after the first one will not. For more information about running multiple Integration Servers on the same machine, see [“Running Multiple Integration Server Instances” on page 63](#).

» **To add an HTTPS diagnostic port**

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Add Port**.
4. Under **Add Port**, select **HTTPS Diagnostic**.
5. Click **Submit**.
6. On the **Edit Diagnostic Port Configuration** screen, under **Diagnostic HTTPS Listener Configuration**, enter the following information:

For this parameter...	Specify...
Enable	Select whether to enable (Yes) or disable (No) this HTTPS diagnostic port.
Port	The number you want to use for the diagnostic port. Select a number that is not already in use on this host machine.
	<p>Note: The <code>watt.server.diagnostic.port</code> server configuration parameter overrides this port number.</p> <p>Important: If you are running multiple Integration Servers on the same host machine, make sure the diagnostic port number on each server is unique.</p>
Alias	An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a-z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.

For this parameter...	Specify...
Package Name	<p>The package associated with this port. The default package is WmRoot. When you enable the package, the server enables the port. When you disable the package, the server disables the port.</p> <p>If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.</p> <p>Note: You cannot change the Package Name associated with this port. The diagnostic port must always be associated with the WmRoot package.</p>
Bind Address (optional)	<p>The IP address to which you want to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use a specific address. If you do not specify a bind address, the server picks one for you.</p>
Backlog	<p>The number of requests that can remain in the queue for an enabled port before Integration Server begins rejecting requests. The default is 200. The maximum value is 65535.</p> <p>Note: This parameter does not apply to disabled ports. Integration Server refuses requests sent to disabled ports.</p>
Keep Alive Timeout	<p>When to close the connection if the server has not received a request from the client within this timeout value (in milliseconds); or when to close the connection if the client has explicitly placed a close request with the server.</p>
Threadpool	<p>Whether the listener will use this pool exclusively for dispatching requests. The existing Integration Server thread pool is a global thread pool. If there is a very high load on this resource, the user may have to wait for the global thread pool to process his request. However, with the private thread pool option enabled, requests coming into this port will not have to compete with other server functions for threads. Click Enable if you wish to employ the private thread pool settings. You can change or accept the default settings given below:</p> <p>Threadpool Min refers to the minimum number of threads for this private threadpool. The default is 1.</p>

For this parameter...	Specify...
	<p>Threadpool Max refers to the maximum number of threads for this private thread pool. The default is 5.</p> <p>Threadpool Priority refers to the Java thread priority. The default is 5.</p> <p>Important: Use this setting with extreme care because it will affect server performance and throughput.</p> <p>If you do not need to use the Threadpool feature, click Disable.</p> <p>When you view the port's details, the server reports the total number of private threadpool threads currently in use for the port.</p>

7. Under **Security Configuration**, enter the following information:

For this parameter...	Specify...						
Client Authentication	<p>The type of client authentication you want Integration Server to perform for requests that arrive on this HTTPS port. See “Authenticating Clients” on page 515 for more information.</p> <p>Select one of the following:</p> <table border="1"> <thead> <tr> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Username/Password</td> <td>Integration Server prompts the client for a user ID and password.</td> </tr> <tr> <td>Digest</td> <td>Integration Server uses password digest for authentication of all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</td> </tr> </tbody> </table> <p>Note: A port that is configured to use password digest for authentication of client requests will process a request from a user only if the user</p>	Option	Description	Username/Password	Integration Server prompts the client for a user ID and password.	Digest	Integration Server uses password digest for authentication of all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.
Option	Description						
Username/Password	Integration Server prompts the client for a user ID and password.						
Digest	Integration Server uses password digest for authentication of all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.						

For this parameter...

Specify...

is configured to allow password digest for authentication. For more information about configuring a user for digest authentication, see [“Adding User Accounts” on page 86.](#)

Request Client Certificates

Integration Server requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. If the client provides a certificate:

- Integration Server checks whether the certificate exactly matches a client certificate that is on file and signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in Integration Server. If not, the client request fails, unless central user management is configured.
- If central user management is configured, Integration Server checks whether the certificate is mapped to a user in the central user database. If so, the server logs the client on as that user. If not, the client request fails.

Require Client Certificates

Integration Server requires client certificates for all requests. The server behaves as described for **Request Client Certificates**, except that the client must always provide a certificate.

Use Identity Provider

Integration Server uses an OpenID Provider to authenticate requests. Integration Server redirects all requests sent to this port to the OpenID Provider specified in **Identity Provider**.

Request Kerberos Ticket

Integration Server looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not

For this parameter...	Specify...
	<p>find the ticket, Integration Server uses user name and password for basic authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p>
	<p>Require Kerberos Ticket</p> <p>Integration Server looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, Integration Server fails the authentication. If the client does not provide any authentication information, Integration Server returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p>
<p>Kerberos Properties(Optional)</p>	<p>Kerberos properties are used to enable Kerberos authentication by providing Kerberos-related details that will be used for handling service requests that come with a Kerberos ticket. For information on configuring Kerberos authentication, see “Kerberos Authentication” on page 517.</p>
	<p>JAAS Context</p> <p>Specify the custom JAAS context used for Kerberos authentication.</p> <p>In the following example, JAAS Context is KerberosClient:</p> <pre data-bbox="982 1627 1458 1837">KerberosClient { com.sun.security.auth.module. Krb5LoginModule required useKeyTab=true</pre>

For this parameter...	Specify...
	<pre>keyTab=alice.keytab; };</pre>
	<p>The <code>is_jaas.cnf</code> file distributed with Integration Server includes a JAAS context named <code>IS_KERBEROS_INBOUND</code> that can be used with inbound requests.</p>
Principal	Specify the name of the principal to use for Kerberos authentication.
Principal Password	<p>Specify the password for the principal that is used to authenticate the principal to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for authorization. The passwords may be encrypted using different encryption algorithms.</p> <p>If the JAAS login context contains <code>useKeyTab=false</code>, you must specify the principal password.</p>
Retype Principal Password	Re-enter the principal password.
Service Principal Name Format	Displays username , which indicates that the principal name of the service is represented as a named user defined in the LDAP or central user directory used for authentication to the KDC.
Service Principal Name	<p>Specify the name of the principal used with the service that the Kerberos client wants to access. Specify the Service Principal Name in the following format:</p> <p><i>principal-name.instance-name@realm-name</i></p>
Use JSSE	<p>If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure Socket Extension (JSSE) socket factory. The default is Yes.</p> <p>If you set this value to No, the port supports only SSL 3.0 and TLS 1.0.</p>

For this parameter...	Specify...
	<p>Note: To control the cipher suites used on Integration Server ports that use JSSE and handle inbound requests, set the <code>watt.net.jsse.server.enabledCipherSuiteList</code>. For more information, see “Server Configuration Parameters” on page 1017.</p>

8. Under **Listener Specific Credentials**, enter the following information:

Note:
Use these settings only if you want to use a different set of credentials from the ones specified on the Certificates Screen.

For this parameter...	Specify...
Keystore Alias	<p>Optional. A user-specified, text identifier for an Integration Server keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p> <p>For more information, see “Configuring Integration Server for Secure Communication” on page 457.</p>
Key Alias	<p>Optional. The alias for the private key, which must be stored in the keystore specified by the above keystore alias.</p>
Truststore Alias	<p>Optional. The alias for the truststore. The truststore must contain the trusted root certificate for the CA that signed Integration Server certificate associated with the key alias. The truststore also contains the list of CA certificates that Integration Server uses to validate the trust relationship.</p>

9. Click **Save Changes**.
10. On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

11. On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Suspending an HTTP/HTTPS Port

By default, Integration Server accepts port connection requests as soon as it receives them. The suspend port setting allows you to stop the port from accepting connections or dispatching more requests.

Note:

If a request is made on the suspended listening port, and the backlog queue is disabled, the listener will not accept any more connections or dispatch any more requests. However, if the backlog queue is enabled and is not full, the connection is queued. If you delete or disable a suspended port, the queued connections are released.

Complete the following steps to suspend an HTTP or HTTPS port.

➤ **To suspend an HTTP or HTTPS port**

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. In the **Port List** table, click **Edit** in the **Advanced** column for the port you want to suspend.
4. Under **Listener Controls**, select the **Suspend** check box.
5. Click **Apply** to save your changes.
6. Click **Return to Ports** to return to the **Security > Ports** screen.

Resuming an HTTP/HTTPS Port

You can resume a port that has been suspended.

➤ **To resume an HTTP or HTTPS port**

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. In the **Port List** table, click **Edit** in the **Advanced** column for the port you want to resume.
4. Under **Listener Controls**, select the **Resume** check box.
5. Click **Apply** to save your changes.
6. Click **Return to Ports** to return to the **Security > Ports** screen.

Testing for HTTPS Requests

To test whether your server is listening to HTTPS requests on the port you specified, bring up your browser and type in `https://localhost:port`.

- If the port is working properly, you will see the logon screen for Integration Server Administrator.
- If Integration Server Administrator does not display, check to see if a service running on the machine is listening to the same port.

Using an FTP/FTPS Port Range

Integration Server provides FTP and FTPS listeners that listen for FTP/FTPS client data connections on any free port. For FTPS, this port usage method requires all ports to be open on the firewall, a situation that firewall administrators prefer to avoid.

You can specify a range of port numbers for the FTP/FTPS listener to use with a client data connection that uses passive transfer mode (PASV). Integration Server provides the following configuration parameters that you can use to specify the lower and upper bounds of the port range:

- `watt.net.ftpPassivePort.min`
- `watt.net.ftpPassivePort.max`

Specifying an FTP/FTPS Port Range

Keep the following operational considerations in mind when setting a port range for FTP and FTPS ports:

- If the `watt.net.ftpPassivePort.min` and `watt.net.ftpPassivePort.max` parameters are not present or undefined, FTP/FTPS listeners continue the previous behavior of listening on any free port.
- If the value specified for `watt.net.ftpPassivePort.min` is less than 1, a default value of 1 is used. If the value specified for `watt.net.ftpPassivePort.max` is greater than 65534, a default value of 65534 is used. When both of these conditions exist simultaneously, FTP/FTPS listeners continue the previous behavior of listening on any free port.
- An error message is returned to the FTP/FTPS client on the command channel when the specified values do not fall within the expected range. For example, if one of the properties is not defined, if the `watt.net.ftpPassivePort.min` value is larger than the `watt.net.ftpPassivePort.max` value, or if one of the properties is not a valid number.
- An error message is also returned when all the ports in the specified port range are in use.
- Specific details of the error messages are available in the `serverYYYYMMDD.log` file.
- You can modify the port range properties in Integration Server Administrator at any time.

Complete the following steps to specify a port range for FTP and FTPS port listeners.

> To specify a port range for FTP and FTPS listeners

1. Start Integration Server and log on to Integration Server Administrator.
2. In Integration Server Administrator, select **Extended** in the **Settings** area of the Navigation panel.
3. If the `watt.net.ftpPassivePort.min` and `watt.net.ftpPassivePort.max` parameters do not appear the **Extended Settings** list, do the following:
 - a. Select **Show and Hide Keys**.
 - b. Select the check boxes next to `watt.net.ftpPassivePort.min` and `watt.net.ftpPassivePort.max`.
 - c. Click **Save Changes**.
4. On the **Settings > Extended** page, click **Edit Extended Settings**.
5. Do the following:

For this extended setting...	Enter this value...
<code>watt.net.ftpPassivePort.min</code>	<i>Minimum_Port_Number</i>
<code>watt.net.ftpPassivePort.max</code>	<i>Maximum_Port_Number</i>

6. Values for *Minimum_Port_Number* and *Maximum_Port_Number* are port numbers from 1 to 65534. When a port range is specified with these properties, only the ports within the specified minimum and maximum port range (inclusive) are used as the listening ports for incoming FTP/FTPS client data connections. *You must specify both a minimum and maximum setting.*
7. Click **Save Changes**.

About the Primary Port

The primary port is an HTTP or HTTPS port that you designate as the main listening port for Integration Server. The server does not reserve the primary port for any special purpose. However, Integration Server will never allow the primary port to be deleted, which guarantees that at least one port is always available.

If a primary port is not specified, Integration Server creates one at start up. By default, Integration Server designates an HTTP port at 5555 as the primary port. If you created multiple Integration Servers on the same machine, you specified a unique primary port number for each server instance during the instance creation process.

The primary port number is also the port number that clients receive when they query your server property `watt.server.port`.

Note:

If you change Integration Server's primary port number after configuring the Broker, the Broker client for your Integration Server may become unsynchronized with your Integration Server's configuration. After changing the primary port, you need to synchronize your Broker clients to the Integration Server's new port configuration. For more information, see [“Synchronizing Broker Clients When the Primary Port for Integration Server Changes”](#) on page 261.

Changing the Primary Port

Keep the following points in mind when designating a port as the primary port:

- The port must be an HTTP or HTTPS port.
- The port must be associated with the WmRoot package.
- The port must be enabled.
- The port must be a standard port. That is, the primary port cannot be a diagnostic port.

➤ To change the primary port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Change Primary Port**.
4. In the **Select New Primary Port** area of the screen, in the **Primary Port** list, select the port you want to make the primary port.

Integration Server Administrator lists only those ports that meet the criteria for being a primary port (enabled standard HTTP or HTTPS ports that are associated with the WmRoot package).

5. Click **Update**.

Deleting a Port

If you no longer need a port, you can delete it.

Important:

You cannot delete the primary port defined for the Integration Server.

➤ To delete a port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.

3. Locate the port in the **Port List**, and click the **X** icon in the **Delete** column. The server displays a dialog box that prompts you to confirm your action. Click **OK** to confirm that you want to delete the port.

Editing a Port

After adding a port, you can edit the port configuration. The port must be disabled before you can edit the configuration.

Note:

You cannot edit the port alias after a port has been created.

> To edit a port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Locate the port you want to edit and click on the port number.
4. Click **Edit < port type > Port Configuration**.
5. Update the information for the port.
6. Click **Save Changes**.

About Enabling/Disabling a Port

If you want to temporarily prevent the server from accepting requests on one of its ports, you can disable that port. This action blocks incoming requests from reaching the server. When a port is disabled, clients receive an error message when they issue requests to it. Later, you can enable the port. If you shut down and restart the server, the port remains disabled until an administrator enables it. Disabling a port is a convenient way to eliminate developer access to an Integration Server once it goes into production.

Another way to enable or disable a port is to enable or disable the package associated with the port. You can associate a package with a specific port so that when you replicate the package, it continues to use a port with the same number on the new server. When a package is associated with a port, enabling the package enables the port and disabling the package disables the port.

Important:

You must leave at least the primary port enabled.

Disabling a Port

Complete the following steps to disable a port.

> To disable a port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Locate the port in the **Port List**, and click the ✓ icon in the **Enabled** column to disable the port. The server displays a dialog box that prompts you to verify your action. Click **OK** to verify you want to disable the port.

The server replaces the ✓ icon with **No** to indicate that the port is now disabled.

When you disable an HTTP/HTTPS port, Integration Server stops accepting new requests immediately. However, it allows all the in-flight requests to complete. Integration Server adds a `Connection: close` header field in the response header for in-flight requests, which indicates that Integration Server will close the connection after the in-flight request is complete.

For HTTP/HTTPS 1.1 requests, Integration Server always adds a `Connection: close` header field in the response header. For HTTP/HTTPS 1.0, Integration Server adds a `Connection: close` header field only when the client includes a `keep-alive` header field in the request.

Enabling a Port

Complete the following steps to enable a port.

> To enable a port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Locate the port in the **Port List**, and click **No** in the **Enabled** column to enable the port. The server displays a dialog box that prompts you to verify your action. Click **OK** to verify you want to enable the port.

The server replaces the **No** with the ✓ icon to indicate that the port is now enabled.

Configuring How Ports Handle Client Certificates

This section describes how to use Integration Server Administrator to view or change how a port handles client certificates. For more information about client certificates, see [“Authenticating Clients” on page 515](#).

> To view or change how a port handles client certificates

1. Open Integration Server Administrator if it is not already open.

2. In the Navigation panel of the screen, on the **Security** menu, click **Ports**.
3. Locate the port whose client certificate settings you want to view, change, or disable (if it is not already disabled).

Note:

To disable a port, click the ✓ icon in the **Enabled** column. The server replaces the ✓ icon with **No** to indicate that the port is now disabled.

4. Click the port number.
5. Click **Edit HTTPS Port Configuration** or **Edit FTPS Port Configuration** to update the information in the fields, as necessary. For field descriptions, see [“Adding an HTTPS Port” on page 162](#) or [“Adding an FTPS Port” on page 175](#), respectively.
6. Click **Save Changes**.
7. Enable the port by clicking **No** in the **Enabled** column.

Adding a Security Provider

If you want to add an HTTPS or FTPS port with a listener that will use a private key and certificate chain residing in a keystore and the keystore is managed by a non standard Security Provider, you may need to add that Security Provider to Integration Server Administrator.

When specifying keystore information in the HTTPS or FTPS port information screen, a non-standard Security Provider may not appear in the KeyStore Type parameter drop-down list. If the Security Provider that you want to use not appear in the list, use the "Add New Security Provider" link to add the Security Provider.

> To add a security provider

1. Open Integration Server Administrator if it is not already open.
2. In the Security menu of the Navigation panel, click **Keystore**.
3. Click **Add Security Provider**.
4. In the **Add Security Provider** area of the screen, in the **Security Provider Class** field, enter the Java class name of the security provider to be used for additional keystore and truststore file types. For example, the name of nCipher's security provider is `com.ncipher.provider.km.nCipherKM`.

Note:

Ensure that the corresponding jar file is in the classpath.

If the keystore type supported by the HSM is not one of the defaults supported by Integration Server, modify the property `watt.security.keyStore.supportedTypes` or `watt.security.trustStore.supportedTypes` respectively to add a new keystore type for the keystore or truststore.

5. Click **Add Provider**.

Integration Server adds the security provider to the list of available security providers. If the keystore type supported by the newly added security provider is one of the default keystore types (JKS, PKCS12) supported by Integration Server and you select that keystore type in the Keystore Type list for an HTTPS or FTPS port, the corresponding provider will be available in the Provider list for that keystore type.

If the keystore type is not supported, modify the properties `watt.security.keyStore.supportedTypes` and `watt.security.trustStore.supportedTypes` to add a new keystore type for the keystore and truststore.

Configuring the Allowed Protocols for JSSE per Port

This section describes how to configure the allowed protocols for JSSE on a per port basis. For more information about allowed protocols for JSSE, see `watt.net.jsse.server.enabledProtocols`.

➤ To configure the allowed protocols for JSSE per port

1. Shut down Integration Server.
2. Open the following file in text editor:

Integration Server_directory /instances/*instanceName*/packages/*packageName*/listeners.cnf

where *instanceName* is the name of the Integration Server instance and *packageName* is the name of the package associated with the port.

3. In the listeners.cnf file, locate the record for the HTTPS and FTPS ports for which you want to specify the allowed protocols.

For example,

- if you want to make changes to an HTTPS port 5333, the port record will start with the following:

```
<record name="HTTPSListener@5333" javaclass="com.wm.util.Values">
```

- if you want to make changes to an FTPS port 4602, the port record will start with the following:

```
<record name="FTPSListener@4602" javaclass="com.wm.util.Values">
```

4. After the `<value name="useJSSE">true</value>` entry in the port record, add the following entry:

```
<value name="jsseEnabledProtocols">SSLprotocols</value>
```

where *SSLprotocols* is a comma-separated list of the SSL protocol versions that the port supports.

For example, to enable TLS 1.1 and TLS 1.2 versions for the port add the following:

```
<value name="jsseEnabledProtocols">SSLv2Hello,TLSv1.1,TLSv1.2</value>
```

Note:

To prevent a protocol downgrade during negotiation, set *SSLprotocols* to a single protocol version that is TLSv1 or higher.

5. Save your changes and close the text editor.
6. Restart Integration Server.

Note:

The `jsseEnabledProtocols` value specified for the port record in the `listeners.cnf` file overrides the value set by `watt.net.jsse.server.enabledProtocols` server configuration parameter.

When the logging facility 0006 Server SSL Interface is set to the Debug logging level, Integration Server writes messages about protocols used for inbound and outbound ports to the server log. At the Trace logging level, Integration Server writes messages about the enabled cipher suites. You can use these server log messages to confirm the enabled protocols for any JSSE port.

Disabling TLS Renegotiation

TLS renegotiation can lead to Denial of Service (DoS) attacks. You can disable TLS renegotiation for all HTTPS and FTPS ports that use JSSE by setting a Java system property. The property that you configure depends on the JSSE provider in the JDK used by Integration Server.

- When using the JSSE provider from Oracle (SunJSSE), set the following Java system property to true to disable TLS renegotiation: `jdk.tls.rejectClientInitiatedRenegotiation`

For more information, see <https://www.oracle.com/technetwork/jp/java/javase/8-compatibility-guide-2156366.html>

- When using the JSSE provider from IBM (IBM JSSE2), use the following Java system property to disable TLS renegotiation: `com.ibm.jsse2.renegotiate`

For more information, see https://www.ibm.com/support/knowledgecenter/en/SSYKE2_7.0.0/com.ibm.java.security.component.70.doc/security-component/jsse2Docs/tlsrenegotiation.html

Note:

HTTPS and FTPS ports that do not use JSSE (the **Use JSSE** property is set to **No**) uses Entrust for SSL which uses the iSaSiLk library. Integration Server ships with iSaSiLk Version 3.03 which does not support disabling renegotiation.

Designating an HTTP/S Port as Stateless

Integration Server now provides the ability to make Integration Server HTTP and HTTPS ports stateless. A stateless port will not provide or maintain any sessions or session IDs for requests received by the port. This might be useful when using multiple non-clustered Integration Servers with a load balancer (this configuration is sometimes referred to as a stateless cluster).

Use the server configuration parameter `watt.server.portStateless` to specify a comma-separated list of the port numbers for the ports on Integration Server that are stateless. For example:

```
watt.server.portStateless =8443
```

```
watt.server.portStateless =8443,8081
```

You can configure any HTTP or HTTPS port on Integration Server to be stateless with the exception of the primary port and the diagnostic port. Additionally, do not configure any ports used for WebSockets or for administering Integration Server to be stateless. If you specify the primary port, diagnostic port, a port that is not an HTTP or HTTPS port, or a port that does not exist in the value for `watt.server.portStateless`, Integration Server logs a warning message stating that the port cannot be made stateless and that it will be skipped.

10 Setting Up the Server Log

■ Overview of the Server Log	214
■ Specifying Amount and Type of Information to Include in the Server Log	215
■ Specifying Whether to Queue Server Log Entries	217
■ Changing the Default Server Log Location	218
■ Configuring the Server Log to Rotate Based on Size	218
■ Limiting the Number of Server Log Files Kept by Integration Server	219
■ Sending Messages About Critical Issues to E-mail Addresses	220
■ Performing Additional Processing on Log Entries	223
■ Viewing the Server Log	223
■ Changing the Log Displays	225
■ Overriding Logging Level and Server Log Location for a Session	227
■ Globalization and the Server Log	228

Overview of the Server Log

The Integration Server server log contains information about operations and errors that occur on Integration Server, such as the starting of Integration Server subsystems and the loading of packages belonging to Integration Server or other webMethods products. Entries are written to the server log by Integration Server's major subsystems, called *facilities*. For example, the Integration Server package facility writes server log entries when it loads and unloads packages, the Integration Server flow manager facility writes server log entries when it processes a flow service, and Integration Server's HTTP ports write server log entries when they receive requests. Entries are also written to the server log by facilities for individual products that are installed on Integration Server, such as Trading Networks Server or adapters. Below is an excerpt from a sample server log.

```

2018-12-04 16:53:37 EST [ISS.0025.0006I] License Manager started
2018-12-04 16:53:38 EST [ISS.0025.0049I] The JCE Unlimited Strength Jurisdiction
Policy File was found
2018-12-04 16:53:38 EST [ISS.0025.0041I] FIPS mode not initialized
2018-12-04 16:53:42 EST [ISS.0025.0024I] JDBC Connection Manager started
2018-12-04 16:53:44 EST [ISS.0095.0013I] Audit Logging initialized.
2018-12-04 16:53:46 EST [BAS.0123.0001I] FSData full consistency check is being
performed. Db:WmRepository 4\RepoV 4
2018-12-04 16:53:46 EST [ISS.0025.0017I] Repository Manager started
2018-12-04 16:53:47 EST [ISS.0025.0021I] ACL Manager started
2018-12-04 16:53:47 EST [ISS.0025.0008I] State Manager started
2018-12-04 16:53:47 EST [ISS.0025.0010I] Service Manager started
2018-12-04 16:53:47 EST [ISS.0025.0020I] Validation Processor started
2018-12-04 16:53:47 EST [ISS.0025.0022I] Statistics Processor started
2018-12-04 16:53:47 EST [ISS.0025.0018I] Invoke Manager started
2018-12-04 16:53:47 EST [ISS.0025.0012I] Cache Manager started
2018-12-04 16:53:47 EST [BAS.0123.0001I] FSData full consistency chec k
is being performed. Db:DocumentStore\ISTransStore
2018-12-04 16:53:47 EST [ISS.0098.0026I] Document Store DefaultStore initialized
2018-12-04 16:53:48 EST [BAS.0123.0001I] FSData full consistency
check is being performed. Db:DocumentStore\TriggerStore
2018-12-04 16:53:48 EST [ISS.0098.0026I] Document Store
VolatileTriggerStore initialized
2018-12-04 16:53:48 EST [ISS.0098.0026I] Document Store TriggerStore
initialized 2018-12-04 16:53:48 EST [BAS.0123.0001I] FSData full
consistency check is being performed. Db:DocumentStore\ISResubmitStore
2018-12-04 16:53:48 EST [ISS.0098.0026I] Document Store ResubmitStore
initialized
2018-12-04 16:53:49 EST [ISS.0153.0000I] webMethods Messaging subsystem started
2018-12-04 16:53:49 EST [ISS.0106.0001I] Join Manager initialized
2018-12-04 16:53:49 EST [ISS.0025.0032I] Dispatcher initialized
2018-12-04 16:53:49 EST [ISS.0134.0128I] JMS subsystem is starting.
2018-12-04 16:53:49 EST [ISS.0134.0998I] DEFAULT_IS_JMS_CONNECTION cache enabled
:: true
2018-12-04 16:53:49 EST [ISS.0134.0000I] JMS subsystem started
2018-12-04 16:53:49 EST [BAS.0123.0001I] FSData full consistency check is being
performed. Db:XASore\DefaultXASore

```

By default, all facilities write to the server log, and the facilities write log entries about critical, error, warning, informational, and debug-related situations. You can have only selected facilities write to the log, and you can increase or decrease the amount of data the facilities provide. This flexibility is useful for troubleshooting. For example, you might temporarily increase the level of detail written to the server log to help uncover the cause of an Integration Server error or

performance problem, and return to a lower level once the problem is resolved. You can also override the amount of information to include in the server log for a particular Integration Server session.

By default, Integration Server queues log entries written by its facilities in memory, then uses a background thread to write them to the server log. You can change that property so that facilities write directly to the server log. Using a background thread improves Integration Server performance, but writing directly causes the entries to appear in the server log more quickly.

Integration Server always writes server log entries to flat files; you cannot store the server log in a database. By default, Integration Server writes server log entries for the current day (defined as midnight to midnight) to the `server.log` file. However, you can configure Integration Server to rotate the `server.log` file by size instead of by day. For more information, see [“Configuring the Server Log to Rotate Based on Size” on page 218](#).

When rotating the server log file, either by day or by size, Integration Server renames the current `server.log` file to have the archive file name and starts a new `server.log` file. The archive file names include the date (*yyyymmdd*) on which the log entries were originally written.

By default, all log and archive files are stored in the *Integration Server_directory* \instances*instance_name*\logs directory. For information about changing the location of the `server.log` file, see [“Changing the Default Server Log Location” on page 218](#).

Integration Server keeps each server log file that it creates. However, to preserve resources, you might want to limit the number of server log files kept by Integration Server. For more information about limiting the number of `server.log` files kept by Integration Server automatically, see [“Limiting the Number of Server Log Files Kept by Integration Server” on page 219](#).

You can override the location of the server log files for a particular session.. For more information about overriding the location of the server log file or the logging levels for a session, see [“Overriding Logging Level and Server Log Location for a Session” on page 227](#).

Specifying Amount and Type of Information to Include in the Server Log

You can specify logging levels for individual products and specific facilities within a product. Because facilities inherit logging levels from the parent product and, in turn, products inherit logging levels from the Default node, you can set the logging level of the Default node to be the default logging level that you want to use for most of the products and facilities. Then, you can change the logging levels for the particular products or facilities for which you want to generate more or less server log information.

By default, all products inherit the logging level of the Default node. Inherited values are shown in gray text. When you explicitly change the logging level for a product or facility, that level overrides the Default node level. Integration Server Administrator displays explicitly set logging levels in **bold** text.

➤ To specify the amount and type of information to include in the server log

1. In Integration Server Administrator, go to the **Settings > Logging** page.

2. In the **Logger List**, click **Server Logger**.
3. Click **Edit Server Logger**.

The **Server Logger Configuration** area lists Integration Server and products that are installed on Integration Server, and the facilities for each of these. To see the facilities and their current logging levels, expand the display by clicking .

4. Do one or more of the following:

Log level	Action
The Default node	Click the level to use in the Logging Level list for the Default node. Integration Server Administrator resets all products and facilities that inherit from the Default node to the new level.
A particular product	Click the level to use in the Logging Level list for the product node. Integration Server Administrator resets all facilities that inherit their values from the product to the new level. If you change a level for a particular product and later want the product to inherit the logging level from the default node again, reset the product logging level to be the logging level of the default node.
A particular facility	Click the level to use in the Logging Level list for the facility. If you change a level for a facility and later want the facility to inherit the logging level from the product again, reset the facility logging level to be the logging level of the product.

For more information about logging levels, see [“Logging Levels” on page 216](#).

Important:

Recording more information consumes more system resources.

5. Click **Save Changes**.

Logging Levels

The logging levels that you can specify are described below. Each logging level includes the indicated type of message plus all messages from the levels above it (for example, the Warn level includes Fatal, Error, and Warn messages).

Level	Integration Server records these entries	Examples
Fatal	Failures that end operations in such a way that the operations cannot successfully continue without user intervention. Failure is <i>very</i> likely to affect other operations or products.	Product cannot read its configuration file and has no default settings
Error	Same as Fatal, except that existing error handling renders the failure unlikely to affect other operations or products.	Business process step failed due to a service error caused by bad input data
Warn	Problems that do not end operations, or unexpected or unusual conditions that might signal impending failure.	Multiple registered JMX servers were discovered when only one is needed
Info	Success of an event that you need to know about.	Package loaded, or connection pool initialized
Debug	Code-level statements recording unusual conditions or decisions that might lead to errors.	Expected an object to be initialized but it is not, or hash table is empty
Trace	Code-level statements recording program flow and state during normal execution.	Method entry/exit, or local and global object state
Off	No information for the product or facility is written to the server log.	

Specifying Whether to Queue Server Log Entries

By default, Integration Server queues log entries written by its facilities in memory, then uses a background thread to write them to the server log. Because only one facility can write to the server log at one time, queuing the entries improves performance by making writing asynchronous. However, if Integration Server shuts down abnormally, all log entries in the queue will be lost. For better quality of service, do not queue the entries. However, each facility will have to wait as long as necessary to write its entry to the server log, so Integration Server performance might be affected.

➤ To specify whether to queue server log entries

1. In Integration Server Administrator, go to the **Settings > Extended** page and click **Show and Hide Keys**. Integration Server Administrator displays a list of the Integration Server configuration properties you can change using this method.
2. Select the check box next to the **watt.server.log.queued** property.
3. Click **Save Changes**. Integration Server Administrator displays the property in the **Extended Settings** box.

4. Click **Edit Extended Settings**.
5. In the **Extended Settings** box, set the property to `true` if you want to queue server log entries or to `false` if you do not want to queue server log entries.
6. Click **Save Changes**.
7. Restart Integration Server.

Note:

To change the number of entries allowed in the server log queue, use the `watt.server.serverlogQueueSize` parameter. The default size is 8192 bytes.

Changing the Default Server Log Location

You can change the default server log location, including the name of the file to which the logging data is written. For example, if you want to save room on the Integration Server host machine, you might change the log file storage location to a directory on a different machine. The `watt.debug.logfile` server configuration parameter controls the location and name of the log file.

By default, the server log location is: *Integration Server_directory*
`\instances\instance_name\logs\server.log`

Note:

For Microservices Runtime, the default server log location is: *Integration Server_directory*
`\logs\server.og`

To change the server log location or file, modify the `watt.debug.logfile` parameter to specify the fully qualified path or relative path to the file to which you want the server to write server log information. The “relative path” is relative to the Integration Server home directory:
Integration Server_directory `\instances\instance_name`

Note:

The Microservices Runtime home directory is : *Integration Server_directory*

You must specify a directory and a filename. You must restart Integration Server for changes to take effect.

Configuring the Server Log to Rotate Based on Size

By default, the Integration Server `server.log` rotates daily at midnight. If services log large payloads, the `server.log` can increase rapidly in size. In addition to consuming resources, a large `server.log` file can be difficult to examine. To avoid this, you can configure Integration Server to rotate the `server.log` file by size in addition to rotating by day.

Integration Server provides a server configuration parameter that you can use to specify the size limit for the server log. When `watt.server.serverlogRotateSize` is set to a valid value, Integration Server rotates the `server.log` when the `server.log` file size reaches that size or at midnight, whichever

occurs first. For example, if `watt.server.serverlogRotateSize` is set to 100KB and at midnight the `server.log` file size is 80KB, Integration Server still rotates the `server.log` file at midnight.

When Integration Server rotates the `server.log` file, Integration Server renames the current log to use the archive file name and starts a new `server.log` file. If rotating based on size, the archive file name uses the format `server.log.yyyyMMdd.n`, where `yyyymmdd` is the date the log file was created and `n` is the number of the log file created that day.

Note:

There is no default value for the `watt.server.serverlogRotateSize` parameter. If no value is specified for `watt.server.serverlogRotateSize`, Integration Server rotates the `server.log` file at midnight only. If an invalid value is specified, Integration Server resets the parameter to -1. Note that a value of -1 results in the same behavior as specifying no value for the parameter.

For more information about the `watt.server.serverlogRotateSize` parameter, review the parameter description in [“Server Configuration Parameters” on page 1017](#).

Limiting the Number of Server Log Files Kept by Integration Server

By default, Integration Server keeps archived server log files indefinitely. Because `server.log` files can rapidly become large or numerous when using more verbose logging levels, you might want to limit the number server log files that Integration Server keeps on the file system.

Integration Server provides the `watt.server.serverlogFilesToKeep` server configuration parameter that you can use to set to limit the number of server log files that Integration Server maintains on the file system, including the current server log file. When Integration Server reaches the limit for the number of server log files, Integration Server deletes the oldest archived server log file each time Integration Server rotates the server log. Following are some examples of values for `watt.server.log.filesToKeep` and the resulting Integration Server behavior:

- If you set `watt.server.serverlogFilesToKeep` to n , Integration Server keeps the current server log file (`server.log`) and up to $n-1$ archived server log files. For example, if you set `watt.server.log.filesToKeep` to 30, Integration Server keeps the current server log file (`server.log`) and up to 29 archived server log files.
- If you set `watt.server.serverlogFilesToKeep` to 1, Integration Server keeps the current `server.log` file and no previous `server.log` files. When Integration Server rotates the `server.log`, Integration Server does not create an archive file for the previous server log.
- If you set `watt.server.serverlogFilesToKeep` to 0, or any value less than 1, Integration Server keeps an unlimited number of server log files.

The default value of `watt.server.serverlogFilesToKeep` is -1, indicating that there is no limit to the number of server log files that Integration Server maintains.

For more information about the `watt.server.serverlogFilesToKeep` parameter, review the parameter description in [“Server Configuration Parameters” on page 1017](#).

Sending Messages About Critical Issues to E-mail Addresses

By default, Integration Server does not send notifications of any log entries to an e-mail address. You can configure Integration Server to automatically send notifications to a specified e-mail address each time a critical issue occurs.

> To send messages about critical issues to e-mail addresses

1. Open Integration Server Administrator if it is not already open.
2. Go to **Settings > Resources**.
3. Click **Edit Resource Settings**.
4. In the **SMTP Server** field, type the server name or IP address of the SMTP server to use to send the messages.
5. In the **Port** field, enter the port that Integration Server is to connect to on the specified SMTP server.
6. From the **Transport Layer Security** list, select the type of SSL encryption that Integration Server uses when communicating with the SMTP server port defined above.

Select	To
None	Default. Use a non-secure mode when communicating with an SMTP server.
Explicit	Integration Server uses explicit security when communicating with an SMTP server.
Implicit	Integration Server uses implicit security when communicating with an SMTP server.

Note:

Microsoft email provider supports only Explicit Transport Layer Security.

7. From the **Truststore Alias** list, select the alias for the truststore that contains the list of certificates that Integration Server uses to validate the trust relationship between Integration Server and the SMTP server. If you do not select a truststore alias, the default truststore alias specified in the `watt.security.trustStoreAlias` property will be used. For more information about this property, see [“watt.security.” on page 1048](#). For more information about truststore alias, see [“Creating Truststore Aliases” on page 482](#).
8. In the **Internal Email** field, type the e-mail address to which to send messages about critical log entries. Typically, you would specify the e-mail address for the Integration Server Administrator.

9. In the **Service Email** field, type the e-mail address to which to send messages about service failures. In a development environment, you might direct these messages to the developer. In a production environment, you might direct these messages to the Integration Server Administrator.
10. By default, Integration Server uses character set UTF-8 for the messages. If you want to use a different character set, identify the character set in the **Default E-mail Charset** field.
11. In the **Username** field, type the e-mail address of the account that Integration Server uses to connect to the SMTP server.
12. In the **Authentication** field, specify the type of authentication that Integration Server should use to connect to the specified e-mail server. Select **Basic Authentication** or **OAuth**.
 - Select **Basic Authentication** if you want to authenticate the user on the specified e-mail server with only login credentials (username and password).
 - Select **OAuth** to authenticate the user on the specified e-mail server using the credentials issued by the OAuth server.

Note:

Only the Authorization Code grant type is supported for OAuth.

Note:

To obtain the OAuth credentials for Integration Server, you must first register Integration Server with the OAuth Server. For details about registering an application and obtaining the OAuth credentials, see your authorization server's documentation.

13. In the **Password** field, type the password associated with the username that identifies you on the SMTP server. This field applies only to Basic Authentication.
14. In the **Auth URL** field, specify the URL of the email provider or endpoint that Integration Server uses to request authorization code. This field applies only to OAuth.
15. In the **Client ID** field, specify the unique public identifier that the OAuth server generates for Integration Server during registration. This field applies only to OAuth.
16. In the **Client Secret** field, specify a unique string that the OAuth server provides to Integration Server. It is known only to Integration Server and the authorization server. This field applies only to OAuth.
17. In the **Scope** field, specify the mail server access permissions configured for Integration Server during registration. You can specify multiple scopes separated by a space. This field applies only to OAuth.
18. In the **Access Token URL** field, specify the URL that Integration Server must use to request an access token from the OAuth server. This field applies only to OAuth.

19. In the **Redirect URL** field, specify the URL that the OAuth server uses to send authentication responses to Integration Server. This field applies only to OAuth.

- If you access Integration Server locally, enter the redirect URL in one of the applicable formats:
 - `http://localhost:{port}/WmRoot/security-oauth-get-authcode.dsp`
 - `https://localhost:{port}/WmRoot/security-oauth-get-authcode.dsp`
- If you access Integration Server remotely, use `https://{ISHostName}:{port}/WmRoot/security-oauth-get-authcode.dsp`

20. Click **Get Authorization Code** to receive a unique string from the OAuth Server that Integration Server requires to request an access token. A new window prompts for user credentials. After a successful login, an authorization code is sent to the redirect URL. Alternatively, you can obtain the authorization code manually in scenarios that do not allow you to obtain the authorization code through Integration Server Administrator.

To obtain the authorization code manually, send a request through a browser in the format: `<URL of an emailprovider>/<TenantID>/oauth2/authorize?response_type=code&prompt=login&redirect_uri=http://host:{port}/WmRoot/securityoauth-get-authcode.dsp&client_id=<clientID>&scope=<scope>`

Here, `response_type` is set to `code` to inform the OAuth server that Integration Server expects an authorization code as the response at the mentioned `redirect_uri`.

Note:

Since the code generated is valid only for one-time usage, get the authorization code each time you update the OAuth relevant fields.

21. **Access Token Expiry Time** field is displayed specifying the time until when an access token issued to Integration Server is valid. This field applies only to OAuth. This is a read-only field displayed based on the access token.

22. **Token Refresh Interval** field specifies the time interval in minutes at which the cron job is scheduled to fetch a new access token. The access token and the refresh token are issued the first time you save the settings for email notifications. Thereafter, the scheduled cron job refreshes and fetches the access token using the refresh token, ensuring that a valid access token is used to send mails. The default is 15 minutes.

Note:

If the access token expires, the user needs to consent again using login credentials and regenerate the tokens by editing the Email Notification Settings.

23. Click **Save Changes** to update the server with the new or changed email notification settings. Your changes take effect immediately, and you start receiving emails in case of any errors. In addition, the Access Token Expiry Time is displayed.

Note:

You can set all the email notification parameters using the new server configuration parameters during Integration Server startup instead of using Integration Server Administrator. If the authentication type is OAuth, Integration Server Administrator generates an access token and a refresh token, and creates the cron job at the startup.

When you set email notification configuration in **Settings > Extended**, a new access token and a refresh token is generated only when there is a authorization code present. A cron job is created if it is not already created during startup.

When sending a message, by default, Integration Server provides its own address (the From Address) as `Integration-Server@localhost`, where `localhost` is the Integration Server host machine. You must provide a valid From Address. Go to **Settings > Extended** in Integration Server Administrator to set the `watt.server.email.from` parameter to the From Address you want the messages to use.

Performing Additional Processing on Log Entries

If you want to perform additional processing on log entries, you can create an event handler. For example, you could create an event handler that sends service log entries to another log, such as the Windows Event Log. For information, see the *webMethods Integration Server Built-In Services Reference* and *webMethods Service Development Help*.

Viewing the Server Log

In Integration Server Administrator, go to the **Logs > Server** page to view the server log. By default, Integration Server uses this format for server log entries:

```
time_stamp time-zone [product_code.logging_facility.message_number.log_level] message_text
```

To see a list of logging facilities, go to the **Settings > Logging** page in Integration Server Administrator. Integration Server gets the time zone value from your JVM.

Note:

In the default message format, the log level is displayed as one character and will be one of the following: C (Fatal), E (Error), W (Warn), I (Info), D (Debug), T (Trace). Your logging level setting determines the types of messages you see in the server log). See [“Specifying Amount and Type of Information to Include in the Server Log” on page 215](#) for more information about logging levels.

The following table shows some product codes:

Code	Meaning
ISU, ISS, ISC, ISP, JBS, BAS, BAT, BAA, BAJ, BAL, BAP, BAR, BAU, BAC, BAB, BAF, BAQ	Internal Integration Server facilities/components
ART	Adapter runtime facilities

Code	Meaning
MNP	Mainframe package
MOD	webMethods Monitor (Facility 119)
MON	webMethods Monitor (Facility 120 Monitor DB)
SAP	SAP Adapter
TNS, TNC	Trading Networks

Using an Alternative Server Log Entry Format

Integration Server offers an alternative server log format that you can switch to:

(logger) [product_code.logging_facility.message_number] time_stamp time_zone log_level message_text

In this format:

- The log level is spelled out and displayed after the time zone. It will be one of the following: Fatal, Error, Warn, Info, Debug, Trace.
- The *logger* field is the name of the logger used by the Java-based log4j logging utility.

➤ To change the format of server log entries

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click Extended.
3. From the **Settings > Extended** page, click **Show and Hide Keys**.

Integration Server Administrator displays a list of the Integration Server configuration properties you can change using this method.

4. Select the check box next to the **watt.debug.layout** property.
5. Click **Save Changes**.
displays the property in the **Extended Settings** box.
6. Click **Edit Extended Settings**.
7. In the **Extended Settings** box, set the property to legacy if you want to use the default format or to new if you want to use the alternative format.
8. Click **Save Changes**.

9. Restart Integration Server.
10. Your logging level setting (see [“Specifying Amount and Type of Information to Include in the Server Log”](#) on page 215) determines the types of messages you see in the server log).

Changing the Log Displays

You can change the display of server log pages in Integration Server Administrator. You can:

- Specify the date and time format to use in log entries.
- Display logged data in different languages.
- Change various aspects of the display permanently.
- Change various aspects of the server log's display temporarily.

Specifying the Date and Time Format to Use in Log Entries

By default, log entries in all logs shown in Integration Server Administrator use the format `yyyy-mm-dd hh:mm:ss`. You can change this format to any other format that is supported by the Java class `java.text.SimpleDateFormat`.

➤ To specify the date and time format to use in log entries

1. In Integration Server Administrator, go to the **Settings > Logging** page.
2. In the **Logger List**, click **Server Logger**.
3. Click **Edit Server Logger**.
4. In the **Format** box, type the date and time format to use. You can specify any format that is supported by the Java class `java.text.SimpleDateFormat` (for example, `yyyy-MM-dd HH:mm:ss z`).
5. Click **Save Changes**.

Note:

You can also change the date and time format for log entries using the `watt.server.dateStampFmt` server configuration parameter. For more information about this parameter, see [“Server Configuration Parameters”](#) on page 1017.

Displaying Logged Data in Different Languages

This section applies only to logged data that is stored in files.

If you want to view logged data in a language other than English, you might have to adjust your text editor or command shell. Integration Server writes the files in the Unicode UTF-8 encoding. These files do not contain a Byte Order Mark (BOM, Unicode character U+FEFF). If the files contain non-ASCII data, such as log entries written in non-U.S. English, you might have to adjust the character encoding used by your text editor or command shell so you can view the log entries.

On a UNIX system, you can adjust the character encoding by changing your locale setting (LC_ALL) to the appropriate UTF-8 encoded locales. For example, to view Japanese characters in a text editor or command shell on a Solaris system, you might change your locale setting to ja_JP.UTF-8.

On a Windows system, because the files do not contain the BOM character, text editors such as Notepad might not detect the UTF-8 encoding correctly. Adjust the encoding manually so you can view the files. To view the logs in the cmd shell, you can use the command `chcp 65001`.

Note:

Changing character encoding will also affect the audit logs, described in the *webMethods Audit Logging Guide*.

Changing the Display Permanently for All Logs

By default, the number of log entries shown for all logs in Integration Server Administrator is 35, and the refresh interval is 90 seconds. You can change these defaults.

Keep the following points in mind when changing the display for the logs:

- Increasing the number of entries displayed significantly can slow system performance.
- Decreasing the refresh interval significantly can slow system performance.
- If you increase the number of displayed entries to a value greater than the threshold established by the `watt.server.log.alertMaxEntries` configuration parameter, Integration Server Administrator displays a warning.
- Changing the log displays will also affect the audit logs, described in the *webMethods Audit Logging Guide*.

➤ To change the display for all logs

1. Open Integration Server Administrator.
2. Navigate to the **Settings > Extended** page and click **Show and Hide Keys**.

Integration Server Administrator displays a list of the Integration Server configuration properties you can change using this method.

3. Select the check box next to the properties you want to change:

If you want to change	Select this property
The default for the Number of entries to display field	watt.server.log.maxEntries
The refresh interval for log entries	watt.server.log.refreshInterval

4. Click **Save Changes**. Integration Server Administrator displays the property in the **Extended Settings** box.
5. Click **Edit Extended Settings**. In the **Extended Settings** box, set each property to a positive integer.
6. Click **Save Changes**. Changes take effect immediately.

If the value that you set for **Number of entries to display** is more than the value specified for `watt.server.log.alertMaxEntries`, every time the log entries are refreshed, Integration Server Administrator displays a message informing that the number of requested entries exceeds the threshold and that displaying more entries might affect Integration Server performance. The message prompts you to confirm displaying the requested number of entries.

Changing the Display Temporarily for the Server Log

To change the display for the server log temporarily, use the **Log display controls** area at the top of the log display page and then click **Refresh**. The changes remain until you change them again, or until you shut down Integration Server, whichever comes first.

Overriding Logging Level and Server Log Location for a Session

You can override the logging level property setting, server log location, or both for a particular Integration Server session by starting Integration Server from the command line with certain options.

➤ To override log level and server log location for a session

1. At a command line, type the following command to switch to the server's home directory:

```
cd Integration Server_directory \profiles\IS_instance_name
```

Note:

The Microservices Runtime home directory is : *Integration Server_directory*

2. Start Integration Server by entering this command:

For Windows: `bin\startup.bat [-debug level] [-log {filename | none | both}]`

For UNIX: `bin/startup.sh [-debug level] [-log {filename | none | both}]`

Options for these commands are shown below.

Option	Description
<i>level</i>	<p>Amount of information to record for all products and facilities listed on the Settings > Logging > View Server Logger Details page for this session. For possible values, see “Logging Levels” on page 216.</p> <p>For this session only, this option overrides the values specified on the Settings > Logging > View Server Logger Details page and on the <code>watt.debug.level</code> property.</p>
<i>filename</i>	<p>Fully qualified or relative path to the file in which to write the server log for this session. Relative path is relative to the Integration Server home directory: <i>Integration Server_directory \instances\instance_name</i></p> <p>Note: The Microservices Runtime home directory is : <i>Integration Server_directory</i></p> <p>The <i>filename</i> must specify a directory and filename.</p> <p>For this session only, the <i>filename</i> option overrides the value specified on the <code>watt.debug.logfile</code> property.</p>
<i>none</i>	<p>Displays the server log on your console (STDOUT).</p> <p>For this session only, the <i>none</i> option overrides the value specified on the <code>watt.debug.logfile</code> property.</p> <p>Note: Server log messages written to STDOUT include the identifier “ISSERVER” to help differentiate server log messages from other messages written to the console.</p>
<i>both</i>	<p>Writes server log information to the computer screen (STDOUT) and the destination specified by the <code>watt.debug.logfile</code> parameter.</p>

Note:

A `-log` switch value of *none* or *both* also determines where Microservices Runtime or an Integration Server equipped with an Microservices Runtime license writes the configuration variables log. Microservices Runtime ignores a *filename* value. If you specify a *filename*, Microservices Runtime writes the configuration variables log file to this location only: *Integration Server_directory /instances/instanceName/logs/configurationvariables.log*.

Globalization and the Server Log

If a webMethods product is equipped with webMethods language packs and some of those language packs correspond to the language used by the operating environment in which the product is running, the product writes its log entries in the language used by the operating system.

If the product is equipped with no language packs or with language packs that do *not* correspond to the language used by the operating system, the product writes its log entries in U.S. English.

Suppose your operating environment uses Japanese as its language. You have installed language packs including the Japanese Language Packs on Integration Server, so Integration Server stores its own log entries in Japanese. You have not installed the Japanese Language packs on Trading Networks, so Integration Server stores Trading Networks' log entries in U.S. English.

Note:

Even if no language packs are installed on the webMethods product and the product is using U.S. English, Integration Server might store log entries from external sources, such as database drivers or adapter resources, in the language used by the operating environment in which the product is running.

11 Configuring Integration Server for webMethods Messaging

■ Overview of webMethods Messaging	232
■ Working with Messaging Connection Aliases	232
■ Authenticating Connections to the Universal Messaging Server	256
■ Specifying the Keep-Alive Mode for the Broker Connection	258
■ Synchronizing Broker Clients When the Primary Port for Integration Server Changes .	261
■ Configuring Document Stores	261
■ Load Balancing with a Non-Clustered Group of Integration Servers	269

11.1 Overview of webMethods Messaging

webMethods messaging is an umbrella term that encompasses the sending and receiving of documents (also called messages) across the webMethods platform. Within the context of Integration Server, webMethods messaging includes the publishing of documents in the native IData format from one Integration Server or an application running on Integration Server to a webMethods messaging provider. The webMethods messaging provider, either Software AG Universal Messaging or webMethods Broker, routes the documents to subscribers on other Integration Servers that retrieve and process the document. webMethods message is also referred to as native messaging.

To use Integration Server to create a publish-and-subscribe solution for webMethods messaging, you need the following components:

- Publishable document types that define the content, structure, and properties of the documents that can be published.
- Services that publish instances of the publishable document types.
- webMethods messaging triggers that subscribe to publishable document types and specify services that processes instances of the publishable document types.
- A webMethods messaging provider that handles the receipt of and routing of documents to subscribers.

To configure Integration Server for webMethods messaging, you need to:

- Create one or more aliases that configure a connection to the webMethods messaging provider. For more information, see [“Working with Messaging Connection Aliases” on page 232](#).
- Configure document stores for holding documents immediately before publication or right after retrieval. For more information, see [“Configuring Document Stores” on page 261](#).
- Create publishable document types, webMethods messaging triggers, and services using the Service Development perspective in Software AG Designer. For more information, see *webMethods Service Development Help*.

Note:webMethods Broker is deprecated.

Working with Messaging Connection Aliases

A messaging connection alias defines the configuration needed to establish a connection between Integration Server and a webMethods messaging provider. You can use Universal Messaging and/or Broker as a webMethods messaging provider. For each messaging provider that you want to use, you need to create a messaging connection alias.

Although an Integration Server can only have one messaging connection alias that connects to a Broker, you can have multiple messaging connection aliases that connect to Universal Messaging servers.

You assign a messaging connection alias to publishable document types. Publishing services use the assigned messaging connection alias to publish instances of that document type to the messaging provider. webMethods messaging triggers use the messaging connection alias to retrieve published documents from the messaging provider.

Predefined Messaging Connection Aliases

Integration Server includes predefined messaging connection aliases that Integration Server creates the first time it starts up.

The following table identifies the predefined messaging connection aliases that might be available for Integration Server.

Messaging Connection Alias	Description
IS_BROKER_CONNECTION	<p>A messaging connection alias that contains the configuration information needed to establish a connection to Broker.</p> <div data-bbox="662 829 1360 1033" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This messaging connection alias only appears if you migrated to Integration Server 10.2 or later from an earlier version of Integration Server that used this alias or configured a connection to Broker.</p> </div>
IS_DES_CONNECTION	<p>A messaging connection alias used to establish a connection with a Universal Messaging server for purposes of sending and receiving events with Digital Event Services. Assign this messaging connection alias to publishable document types that will be published as events to Digital Event Services.</p> <p>You cannot delete the IS_DES_CONNECTION alias.</p>
IS_LOCAL_CONNECTION	<p>A messaging connection alias to use with document types that will be published locally only. In local publishing, a document is published and received within Integration Server. In local publishing, the document remains within Integration Server. There is no involvement with a webMethods messaging provider.</p> <div data-bbox="662 1564 1360 1873" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If you migrate to Integration Server version 9.9 or later and you are migrating an Integration Server that did not specify a default connection alias, Integration Server assigns the IS_LOCAL_CONNECTION alias to any publishable document type that specifies DEFAULT or does not specify a messaging connection alias.</p> </div>

Messaging Connection Alias	Description
IS_UM_CONNECTION	A messaging connection alias that contains the configuration information needed to establish a connection to a Universal Messaging server.

Creating a Broker Connection Alias

A Broker connection alias is a webMethods messaging connection alias that contains the configuration information for establishing a connection to the Broker. Each Integration Server can have only one Broker connection alias.

Note: Integration Server version 10.5 works in the same manner when connected to Broker 9.6 or Broker 10.5.

If you migrated to Integration Server 10.2 or later from an earlier version of Integration Server, a Broker connection alias named IS_BROKER_CONNECTION might already exist. Or, if you migrated to Integration Server 9.5 SP1 or later from an earlier version of Integration Server, and the earlier Integration Server configured a connection to Broker, the existing Broker configuration information is used for the IS_BROKER_CONNECTION alias. If this alias exists, you must delete it before you create another Broker connection alias. Alternatively, you can edit IS_BROKER_CONNECTION to contain the configuration information that you want.

Note: webMethods Broker is deprecated.

➤ To create a Broker connection alias

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. Click **Create Broker connection alias**.

If a Broker connection alias already exists, Integration Server Administrator displays a message stating that only one Broker connection alias can exist at a time.

5. Under **General Settings**, specify the following:

Field	Description
Connection Alias Name	A unique name for the messaging connection alias.
Description	A description of the messaging connection alias.

Field	Description
Client Prefix	<p>A string that identifies the Integration Server to Broker.</p> <p>The Broker Manager displays this prefix for each client it creates for the server. (The Broker creates multiple clients for each server that connects to it.)</p>
Client Prefix Is Shared (prevents removal of shared messaging provider objects)	<p>Whether Integration Server shares the client prefix with multiple Integration Servers and you want to prevent automatic updates, including deletions, to shared objects associated with this alias on the Broker. Leave the Client Prefix Is Shared check box cleared if you do not want to prevent automatic updates to shared objects on the Broker.</p> <p>When the Client Prefix Is Shared check box is selected, you must update objects on the Broker manually. For example, if you delete a trigger on Integration Server you need to manually delete the associated client queue on the Broker.</p> <p>You must select Client Prefix Is Shared if the Integration Server:</p> <ul style="list-style-type: none"> ■ Belongs to a cluster and you want to prevent automatic updates to shared objects associated with this alias on the Broker. ■ Belongs to a non-clustered group of Integration Servers that operate in a load-balanced fashion and you want to prevent automatic updates to shared objects associated with this alias on the Broker. <p>For more information about configuring Integration Servers to receive messages from Broker in a load-balanced fashion, see “Load Balancing with a Non-Clustered Group of Integration Servers” on page 269.</p>

6. Under **Connection Settings**, specify the following:

Field	Description
Broker Host	Name (<i>DNSname: port</i> or <i>ipaddress: port</i>) of the machine on which the Broker Server resides.
Broker Name	Name of the Broker as defined on the Broker Server. The default name is Broker #1 .
Client Group	Broker client group to which this Integration Server belongs. A client group defines a single set of properties and access permissions assigned to one or more clients (here, Integration Servers) on a single Broker. If the specified client group does not

Field	Description
	exist, the Integration Server creates it on the named Broker upon establishing its initial connection.

Important:Brokers do not share can-publish and can-subscribe permissions across client groups. If you switch an Integration Server from one client group to another, you must restart the Integration Server and synchronize all publishable document types with the Broker. Next, you must shut down the server and use My webMethods to delete all of the Broker clients created for the server with the changed client group. Restart the server with the changed client group.

7. Under **Client Authentication Settings**, specify the following:

Field	Description
Client Authentication	<p>The type of authentication an Integration Server client will use to connect to the Broker. Choose one of the following options:</p> <ul style="list-style-type: none"> ■ None. Select this option if the Broker is configured to accept anonymous connections. ■ Username/Password. Select this option if the Broker uses basic client authentication. If you select this option, specify the user name and password the client will use in the Username and Password fields. ■ SSL. Select this option if Integration Server connects to the Broker using the SSL port. If you select this option, configure the SSL parameters by providing the following information.

Field	Description
Keystore	<p>The full path to the keystore file for this Integration Server. A keystore file contains the credentials (private key/signed certificate) that an entity needs for SSL authentication. If the Broker Server requires an SSL connection, then the information in this file is used to authenticate the Integration Server client to that Broker Server.</p> <p>The keystore file for an Integration Server is stored on the machine on which the Integration Server resides.</p>

Field	Description
Keystore Type	The file type of the keystore file for Integration Server. The file type can be either PKCS12 or JKS .
Keystore Password	Password required to access the SSL certificate in the keystore file for Integration Server.

8. Under **Encryption Settings**, specify the following:

Field	Description
Encryption	Specify whether or not to encrypt the connection between the Integration Server and the Broker.

Note:

When you set **Client Authentication** to **SSL**, **Encryption** must be set to **Yes**.

If you select **Yes** for the **Encryption** parameter, configure the following truststore parameters.

Field	Description
Truststore	The full path to the truststore file for the Integration Server client. A truststore file contains trusted root certificates for the authorities responsible for signing SSL certificates. For an SSL connection to be made, a valid trusted root for the SSL certificate stored in the keystore must be accessible in a truststore file. The truststore file for Integration Server is stored on the machine on which the Integration Server resides.
Truststore Type	The file type of the truststore file for Integration Server. The file type is JKS .

9. Click **Save Changes**.

Integration Server creates the Broker connection alias.

10. Click **Return to webMethods Messaging Settings**.

11. Enable the Broker connection alias.

12. Restart Integration Server.

For more information about Broker and configuring SSL for Broker, see *Administering webMethods Broker*.

Note:

You can configure the outbound document store, also known as the client side queue, to contain documents published when the Broker used by this connection alias is not available. For more information, see [“About the Outbound Document Store” on page 266](#)

Creating a Universal Messaging Connection Alias

A Universal Messaging connection alias is a webMethods messaging connection alias that contains the configuration information for establishing a connection to a Universal Messaging server. Each Integration Server can have multiple Universal Messaging connection aliases.

If Integration Server uses the Universal Messaging connection alias to connect to a secure socket protocol port on the Universal Messaging server, you must configure the connection alias to use SSL. You specify truststore and possibly keystore information in the alias. This allows communication between Integration Server and Universal Messaging to be secured through secure socket layers (SSL). A Universal Messaging port uses SSL if the port specifies an interface protocol of NSPS or NHPS. Depending on how the Universal Messaging port is configured, you might need to supply a client certificate:

- If the Universal Messaging port specifies NSPS or NHPS and the **Enable Client Cert Validation** option is *not* selected for the port, you need to specify a truststore alias during configuration of the Universal Messaging connection alias.
- If the Universal Messaging port specifies NSPS or NHPS and the **Enable Client Cert Validation** option *is* selected for the port, you need to specify a truststore alias, keystore alias, and key alias during configuration of the Universal Messaging connection alias.

Note:

Use Universal Messaging Enterprise Manager to view the protocol and **Enable Client Cert Validation** option selected for the port.

If the Universal Messaging connection alias establishes a connection to a secure port on Universal Messaging, you need to do the following before completing configuration of the alias.

- Create a truststore that contains the certificate authority (CA) of the certificates for the Universal Messaging server. Then create a truststore alias for the truststore. For more information about creating a truststore alias, see [“Creating Truststore Aliases” on page 482](#).
- If the Universal Messaging port performs client certificate validation, do the following:
 - Create a keystore that contains the client certificates used by Integration Server to connect with Universal Messaging. Create a keystore alias for the keystore, making sure to specify a key alias for the key that contains the private key for connecting to the Universal Messaging port securely. For more information about creating a keystore alias, see [“Creating Keystore Aliases” on page 481](#).

- Verify that truststore used by Universal Messaging contains the certificate authority of the certificates used by Integration Server.

Note:

Keep the Universal Messaging client libraries up to date. If you install a Universal Messaging fix that updates the client libraries, make sure to copy the updated Universal Messaging client library files to the *Software AG_directory /common/lib* directory used by Integration Server.

Use the following procedure to create a Universal Messaging connection alias.

➤ **To create a Universal Messaging connection alias**

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. Click **Create Universal Messaging connection alias**.
5. Under **General Settings**, specify the following:

Field	Description
Connection Alias Name	A unique name for the messaging connection alias.
Description	A description of the messaging connection alias.
Client Prefix	A string that identifies the Integration Server to Universal Messaging. If your Integration Server belongs to a cluster, make sure all servers in the cluster use the same client prefix.
Client Prefix Is Shared (prevents removal of shared messaging provider objects)	Whether Integration Server shares the client prefix with multiple Integration Servers and you want to prevent automatic updates, including deletions, to shared objects associated with this alias on Universal Messaging server. Leave the check box cleared if you do not want to prevent automatic updates to shared objects on the Universal Messaging server. When the Client Prefix Is Shared check box is selected, you must update objects on Universal Messaging manually. For example, if you delete a trigger on Integration Server you need to manually delete the associated channel on Universal Messaging. You must select Client Prefix Is Shared if the Integration Server:

Field	Description
	<ul style="list-style-type: none">■ Belongs to a cluster and you want to prevent automatic updates to shared objects associated with this alias on the Universal Messaging server.■ Belongs to a non-clustered group of Integration Servers that operate in a load-balanced fashion and you want to prevent automatic updates to shared objects associated with this alias on the Universal Messaging server. <p>For more information about configuring Integration Servers to receive messages from Universal Messaging in a load-balanced fashion, see “Load Balancing with a Non-Clustered Group of Integration Servers” on page 269.</p> <ul style="list-style-type: none">■ The Universal Messaging connection alias is used by loggers on multiple Integration Servers in a cluster or non-clustered group to write or read logs from a Universal Messaging queue. Selecting the Client Prefix Is Shared check box ensures that Universal Messaging queue name includes the client prefix which allows Integration Servers to write to and read from the same logging queue across a Universal Messaging realm. For more information about using a Universal Messaging queue as the logging queue, see the <i>webMethods Audit Logging Guide</i>.

6. Under **Connection Settings**, specify the following:

Field	Description
Realm URL	<p>The URL for the Universal Messaging server in the format <i>protocol://UM_host:UM_port</i> where <i>protocol</i> is the protocol for the port, such as <i>nsp</i>, <i>nsps</i>, or <i>nhps</i>. For example, <i>nsp://127.0.0.1:9000</i> or <i>nsps://localhost:9000</i>.</p> <p>If the protocol for connecting to Universal Messaging server is <i>nsps</i> or <i>nhps</i>, you must provide the appropriate certificates for use with the connection alias. You must provide a truststore alias or a truststore alias and keystore alias.</p> <p>If you are using a cluster of servers, supply the list of URLs to each server in the cluster using a comma-separated or a semicolon-separated list. By default, Integration Server is configured to follow the master realm server in a cluster, which indicates that Integration Server always connects to the master realm server in the cluster. Software AG recommends using a comma-separated list of URLs if the alias uses follow the master.</p>

Field	Description
	<p>Note: You must specify a URL for <i>every</i> Universal Messaging server in the cluster. Integration Server connects only to the Universal Messaging servers for which you specify a URL. Consequently, Integration Server creates resources such as channels and durable subscriptions only on the specified Universal Messaging servers. If you specify only one Universal Messaging server in a cluster, the resources will be local to that server only.</p>
	<p>To use the Universal Messaging horizontal scalability feature with this messaging connection alias, enclose each realm URL or multiple realm URLs separated by commas in parenthesis. In a cluster, the realm URLs must be separated by a comma.</p>
	<p>➤ Examples of Horizontal Scalability URL Syntax</p>
	<ul style="list-style-type: none"> ■ <i>(realm1)(realm2)(realm3)(realm4)</i> - Indicates 4 standalone realms, namely <i>realm1</i>, <i>realm2</i>, <i>realm3</i> and <i>realm4</i>. ■ <i>(realm1,realm2)(realm3,realm4)</i> - Indicates 2 clusters, one consisting of <i>realm1</i> and <i>realm2</i> and the other consisting of <i>realm3</i> and <i>realm4</i>. ■ <i>(realm1)(realm2,realm3)(realm4)</i> - Indicates one cluster consisting of <i>realm2</i> and <i>realm3</i>, and two standalone realms, namely <i>realm1</i> and <i>realm4</i>.
	<p>If you disabled the follow the master behavior for producers and/or consumers using this alias, whether you separate the realm servers in a cluster with a comma or a semicolon determines the Universal Messaging realm server to which Integration Server connects.</p>
	<ul style="list-style-type: none"> ■ Separate the URLs using a comma if Integration Server always attempts to connect to the first Universal Messaging server in the list, only trying the second Universal Messaging server in the list if the first server becomes unavailable. ■ Separate the URLs using a semicolon if Integration Server connects to a randomly chosen URL from the list. This may result in better distribution of clients across the servers in the cluster.
	<p>For more information about the follow the master behavior including how to disable it, see “About Follow the Master for webMethods Messaging” on page 248.</p>

Field	Description
Maximum Reconnection Attempts	Specify the maximum number of reconnection attempts that Integration Server makes if the connection to Universal Messaging fails. If the connection cannot be re-established, Integration Server writes messages to the error log and the connection created by this messaging connection alias will be stopped. The default is 5 attempts.

Note:

The **Maximum Reconnection Attempts** value is a factor only when an existing connection fails. The **Maximum Reconnection Attempts** value is not a factor that Integration Server considers when starting a messaging connection alias.

7. Under **Producer Settings**, specify the following:

Field	Description
Enable CSQ	Whether a client side queue is used with this Universal Messaging connection alias. Do one of the following: <ul style="list-style-type: none">■ Select the Enable CSQ check box if you want to use a client side queue with this Universal Messaging connection alias. When a client side queue is in use, if the Universal Messaging server is not available when Integration Server publishes documents using this Universal Messaging connection alias Integration Server writes the documents to the client side queue.■ Clear the Enable CSQ check box if you do not want to use a client side queue with this Universal Messaging connection alias. When a client side queue is not in use, if the Universal Messaging server is not available when Integration Server publishes documents using this Universal Messaging connection alias, the publishing service ends with an <code>ISRuntimeException</code>.
Maximum CSQ Size (messages)	The maximum number of documents (messages) that can exist in the client side queue for this Universal Messaging connection alias. If the client side queue is at capacity, publishing services that use this connection alias will end with an <code>ISRuntimeException</code> .

Field	Description
Drain CSQ in Order	<p>Specify -1 if you want the client side queue to be able to contain an unlimited number of messages.</p> <p>Whether Integration Server drains the client side queue for this alias by sending the messages to the Universal Messaging server in the same order in which Integration Server placed the messages in the client side queue. Do one of the following:</p> <ul style="list-style-type: none"> ■ Select the Drain CSQ in Order check box if you want Integration Server to send messages to the Universal Messaging in publication order when the connection to the Universal Messaging server is re-established. After the connection to the Universal Messaging server is re-established, Integration Server continues to write newly published messages to the client side queue until the client side queue is completely drained. ■ Clear the Drain CSQ in Order check box if you do not want Integration Server to preserve publication order when the connection to the Universal Messaging server is re-established. After the connection to the Universal Messaging server re-established, Integration Server sends newly published messages directly to the Universal Messaging server while draining the client side queue.
Publish Wait Time While Reconnecting	<p>The number of milliseconds that a publishing service using this Universal Messaging connection alias will wait for a connection to the Universal Messaging server to be re-established after the connection fails. If Integration Server re-establishes the connection before the Publish Wait Time While Reconnecting elapses, the publishing service continues executing. If the specified time elapses before a connection is re-established, the publishing service ends with an <code>ISRuntimeException</code>. The default is 0 milliseconds, which means that publishing services will not wait for Integration Server to re-establish the connection.</p>

Note:

Field	Description
	<p>When a Universal Messaging connection alias is configured to connect to a Universal Messaging cluster, make sure the Publish Wait Time While Reconnecting value is long enough to accommodate the time the Universal Messaging cluster takes to establish a quorum after one of the member servers fails.</p>
<p>Enable Follow the Master for Producers</p>	<p>Whether Integration Server always connects to the master realm server in a Universal Messaging cluster when Integration Server uses this Universal Messaging connection alias to publish messages. Do one of the following:</p> <ul style="list-style-type: none"> ■ Select the Enable Follow the Master for Producers check box to indicate that Integration Server always connects to the master realm server in a Universal Messaging cluster when Integration Server uses this Universal Messaging connection alias to publish messages. ■ Clear the Enable Follow the Master for Producers check box to disable the follow the master behavior for this alias when it is used to send messages. <p>When follow the master is disabled for producers, Integration Server connects to a server in a Universal Messaging cluster using the order dictated by the comma or semi-colon separated list of URLs in the Realm URL parameter.</p> <p>For more information about the follow the master behavior, see “About Follow the Master for webMethods Messaging” on page 248.</p>

8. Under **Consumer Settings**, specify the following for the **Enable Request/Reply Channel and Listener** check box.

Field	Description
<p>Enable Request/Reply Channel and Listener</p>	<p>Whether you intend to use this Universal Messaging connection alias to send request documents, send reply documents, or receive reply documents as part of a request/reply or publish-and-wait scenario. Do one of the following:</p>

Field	Description
	<ul style="list-style-type: none"> <li data-bbox="690 262 1341 472">■ Select the Enable Request/Reply Channel and Listener if you intend to use this Universal Messaging connection alias to send request documents, send reply documents, or receive reply documents as part of a request/reply or publish-and-wait scenario <li data-bbox="690 493 1341 703">■ Clear the Enable Request/Reply Channel and Listener check box if you do not intend to use this Universal Messaging connection alias as part of a request/reply or publish-and-wait scenario. This will conserve resources on Integration Server and Universal Messaging. <p data-bbox="690 724 1341 871">When the Enable Request/Reply Channel and Listener check box is selected, Integration Server does the following when the Universal Messaging connection alias starts:</p> <ul style="list-style-type: none"> <li data-bbox="690 892 1341 1081">■ Create a request/reply channel for this Universal Messaging connection alias if one does not yet exist. The request/reply channel name includes a string identifying the Integration Server, the client prefix, and the text “RequestReply”. <li data-bbox="690 1102 1341 1197">■ Start a listener on Integration Server that subscribes to the alias-specific request/reply channel.
Enable Follow the Master for Consumers	<p data-bbox="690 1218 1341 1396">Whether Integration Server always connects to the master realm server in a Universal Messaging cluster when Integration Server uses this Universal Messaging connection alias to retrieve messages. Do one of the following:</p> <ul style="list-style-type: none"> <li data-bbox="690 1417 1341 1627">■ Select the Enable Follow the Master for Consumers check box to indicate that t Integration Server always connects to the master realm server in a Universal Messaging cluster when using this Universal Messaging connection alias to retrieve messages <li data-bbox="690 1648 1341 1795">■ Clear the Enable Follow the Master for Consumers check box to disable the follow the master behavior for this alias when it is used to retrieve messages. <p data-bbox="690 1816 1341 1885">When follow the master is disabled for consumers that use this alias, Integration Server connects to a</p>

Field	Description
	<p>server in a Universal Messaging cluster using the order dictated by the comma or semi-colon separated list of URLs in the Realm URL parameter.</p> <p>For more information about the follow the master behavior, see “About Follow the Master for webMethods Messaging” on page 248.</p>

9. Under **Client Authentication Settings**, specify the following:

Field	Description
Client Authentication	<p>The type of authentication an Integration Server client will use to connect to the Universal Messaging server. Choose one of the following options:</p> <ul style="list-style-type: none"> ■ None. Select this option if no authentication should be performed. This is the default. <p>Even if Universal Messaging does not perform client authentication, if the protocol specified for Realm URL is nspcs or nhps, you must select a truststore alias in the Truststore Alias field.</p> ■ Username/Password. Select this option if the Universal Messaging server performs basic client authentication using user name/password combinations. If you select this option, specify the user name and password the client will use in the Username and Password fields. <p>In addition to providing the username and password for client authentication, if the protocol specified for Realm URL is nspcs or nhps, you must also select a truststore alias in the Truststore Alias field.</p> ■ Certificate Based. Select this option if Integration Server connects to a secure socket layer (SSL) port on the Universal Messaging and Universal Messaging performs certificate-based authentication. <p>Depending on how the Universal Messaging port is configured, you must specify a truststore alias and possibly a keystore alias and key alias for the Universal Messaging connection alias.</p> <ul style="list-style-type: none"> ■ If the Universal Messaging port does not have the Enable Client Cert Validation option selected, you must specify a truststore alias. ■ If the Universal Messaging port has the Enable Client Cert Validation option selected, you must specify a truststore alias, keystore alias, and key alias.

Note:

Field	Description
	Use Universal Messaging Enterprise Manager to view the protocol and Enable Client Cert Validation option specified for a port.
Truststore Alias	<p>The alias for the truststore that contains the certificate authority (CA) certificates for the Universal Messaging server.</p> <p>You must select a truststore alias if the Universal Messaging port is configured for certificate-based authentication or if the protocol specified for Realm URL is <code>nsps</code> or <code>nhps</code>.</p>
Keystore Alias	<p>The alias for the keystore that contains the client certificates that you want Integration Server to use when connecting to the Universal Messaging port. You must select a keystore alias if the Universal Messaging port is configured for certificate based authentication.</p>
Key Alias	<p>The alias to the key that contains the private key for connecting to the Universal Messaging port securely. The key alias must be in the keystore specified in Keystore Alias.</p> <p>You must specify a Key Alias if you specify a Keystore Alias.</p>

10. Under **Enhanced Logging**, specify the following to configure additional logging for messages sent and/or received using this connection alias:

Field	Description
Logging Type	<p>Specifies where Integration Server writes log messages when enhanced logging is enabled for the message producers and/or consumers that use this messaging connection alias to send and/or receive documents.</p> <ul style="list-style-type: none"> ■ SERVER LOG. Select this option to write enhanced logging messages to the server log. If you specify the server log as the destination, make sure to increase the logging level for the 0168 Messaging (Enhanced Logging) server log facility to at least Info. ■ MESSAGING AUDIT LOG. Write enhanced logging messages to the messaging audit log. <p>You can select one of the options only. Integration Server cannot write enhanced logging messages to the server log <i>and</i> the messaging audit log.</p>
Enable Producer Message ID Tracking	<p>Select to indicate that Integration Server writes additional log messages when a message producer uses this messaging connection alias to publish documents that are instances of the publishable</p>

Field	Description
	document types listed in Producer Message ID Tracking: Include Channels .
Producer Message ID Tracking: Include Channels	The name of each channel that corresponds to a publishable document type for which Integration Server performs additional logging during publication. Use a semicolon (;) to separate each channel name. Leave this field blank if you want Integration Server to perform enhanced logging for every channel to which this messaging connection alias sends messages
Enable Consumer Message ID Tracking	Select to indicate that Integration Server writes additional log messages for messaging consumers (triggers) that use this messaging connection alias to receive messages. Integration Server writes additional log message for the triggers listed in Consumer Message ID Tracking: Include Triggers .
Consumer Message ID Tracking: Include Triggers	The fully qualified name of the triggers for which Integration Server performs additional logging during trigger processing. Use a semicolon (;) to separate each trigger name. Leave this field blank if you want Integration Server to perform enhanced logging for every trigger that uses this messaging connection alias to receive messages.

For more information about using enhanced logging for documents sent to and received from Universal Messaging, [“Using Enhanced Logging for Messaging” on page 335](#).

11. Click **Save Changes**.

12. Enable the Universal Messaging connection alias.

Note:

To configure a Universal Messaging connection alias to connect to a Universal Messaging port that performs certificate-based authentication, the alias needs to specify a truststore alias, keystore alias, and key alias. However, if you set **Client Authentication** to **Certificate Based** in the connection alias but do not supply the truststore alias or keystore alias needed to establish an SSL connection with the Universal Messaging server, Integration Server looks for this information in the JVM. For information about setting the SSL-related system properties in the JVM, see [“Storing SSL Information for the Integration Server JVM in a Secure Manner” on page 464](#).

Software AG recommends specifying the truststore alias, keystore alias, and key alias information in the Universal Messaging connection alias instead of relying on the JVM system properties.

About Follow the Master for webMethods Messaging

Follow the master is a Universal Messaging client setting that indicates that the client session always connects to the master realm server in the Universal Messaging cluster. If the master realm server is not available, Integration Server does not establish a connection. For a Universal Messaging

connection alias, you can control whether a client session created from the alias follows the master for producer connections, consumer connections, both, or neither.

In many cases, follow the master can offer performance benefits. By default, each Universal Messaging connection alias is configured to follow the master for producer and consumer connections. Connections created for services that use the Universal Messaging connection alias to publish messages will follow the master realm server. webMethods messaging triggers that use the Universal Messaging connection alias to receive messages will use a connection that follows the master realm server.

However, there may be situations for which you do not want the producer and/or consumer connections to follow the master realm server. For example, if your solution has a large number of webMethods messaging triggers using many connections, it might provide better performance to distribute the load across the Universal Messaging cluster. In this case, disable follow the master for consumers using the Universal Messaging connection alias.

For a Universal Messaging connection alias, Integration Server provides a producer option and a consumer option for indicating whether client sessions created from the alias follows the master.

- For message producers, the **Enable Follow the Master for Producers** check box determines whether the client session established for the message producer always connects to the master realm server.
- For message consumers, the **Enable Follow the Master for Consumers** check box determines whether the client session established for the message consumer, such as a webMethods messaging trigger, always connects to the master realm server.

When follow the master behavior is disabled for producers and/or consumers using a Universal Messaging connection alias, whether you separate the realm servers listed in the **Realm URL** parameter with a comma or a semicolon determines the Universal Messaging realm server to which Integration Server connects. For more information on the **Realm URL** parameter, see [“Creating a Universal Messaging Connection Alias” on page 238](#).

Note:

- Prior to Integration Server 10.0, the only way to disable follow the master behavior for webMethods messaging was to modify the `custom_wrapper.conf` file to include the `wrapper.java.additional.n=-DFollowTheMaster=false` where `n` was the next available `wrapper.java.additional` number. Do not use the `DFollowTheMaster` parameter to control the follow the master behavior for Integration Server. This parameter setting will override the follow the master behavior set in Universal Messaging connection aliases and JMS connection aliases.
- If the `Delete Universal Messaging Durable` property in webMethods Deployer is set to **Yes**, Integration Server removes the durable objects from Universal Messaging. For more information, see the *webMethods Deployer User's Guide*.

For information about configuring follow the master for JMS connection aliases that use Universal Messaging as the JMS provider, see the description of the **Enable Follow the Master** option in [“Creating a JMS Connection Alias” on page 285](#).

Editing a Messaging Connection Alias

You can edit the properties of any messaging connection alias that you create or some of the default messaging connection aliases created by Integration Server. Keep the following points in mind before editing a messaging connection alias:

- You can edit any properties of an existing messaging connection alias with the exception of the connection alias name.
- You must disable a messaging connection alias before you can edit it. If the alias is not disabled, the *Edit messagingConnectionAlias* name link will not be hyper-linked.
- If you change the client prefix for an existing Universal Messaging connection alias, Integration Server creates new durable objects on Universal Messaging for triggers that use the alias. The durable objects that use the old client prefix are not automatically removed from Universal Messaging. Old durable objects might contain messages and continue to receive messages.
- For the IS_DES_CONNECTION alias, you can edit the description and client prefix only. If the Integration Server is part of a cluster (stateless or stateful), the client prefix for the IS_DES_CONNECTION connection alias needs to be the same on each Integration Server.
- You cannot edit the IS_LOCAL_CONNECTION alias.

> To edit a messaging connection alias

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. In the **webMethods Connection Alias Definitions** table, click the name of the messaging connection alias that you want to edit.
5. Do one of the following depending on whether you are editing a Broker connection alias, Universal Messaging, or Digital Event Services connection alias:
 - Click **Edit Broker Messaging Connection Alias**.
 - Click **Edit Universal Messaging Connection Alias**.
 - Click **Edit Digital Event Services Connection Alias**.

Note:

If the *Edit messagingConnectionAliasType* link appears but is not available, that is, it is not hyper-linked, the messaging connection alias might not be disabled. You can only edit a disabled messaging connection alias.

6. Update the information for the messaging connection alias.

7. Click **Save Changes**.
8. Enable the messaging connection alias. For more information about enabling a messaging connection alias, see [“Enabling a Messaging Connection Alias” on page 251](#).
9. If this is a Broker connection alias, you must restart Integration Server for your changes to take effect.
10. If this is a Universal Messaging connection alias used for writing or reading logging entries to a Universal Messaging queue, you must restart Integration Server for changes to take effect.

Notes:

- If you changed a Broker connection alias to connect to Broker in another territory, you might need to synchronize your publishable document types with the new Broker. For information about synchronizing publishable document types, see *webMethods Service Development Help*.
- If you change the client prefix of a Universal Messaging connection alias for which the **Enable Request/Reply Channel and Listener** check box is selected, Universal Messaging creates a new request/reply channel with the new client prefix. However, Universal Messaging does not delete the previous request/reply channel for the Universal Messaging connection alias. Use Universal Messaging Enterprise Manager to delete the old channel.
- If you edit a Universal Messaging connection alias by clearing the **Enable Request/Reply Channel and Listener** check box, the request/reply channel for Universal Messaging connection alias remains on the Universal Messaging server. If you want to delete the channel, use Universal Messaging Enterprise Manager to delete the channel.
- For a Universal Messaging connection alias used by loggers to write log entries to or read log entries from a Universal Messaging queue, the state of the **Client Prefix Is Shared** check box determines whether or not the client prefix of the alias is included in the Universal Messaging queue name. If you edit a Universal Messaging connection alias by selecting or clearing the **Client Prefix Is Shared** check box and the alias is used by loggers, you must restart Integration Server for changes to the queue name to take effect. Additionally, you must make the same changes to other Integration Servers in the cluster or non-clustered group of Integration Servers that use the alias.

Enabling a Messaging Connection Alias

When a messaging connection alias is enabled, Integration Server can use it to send and receive messages from the messaging provider on behalf of publishing services and webMethods messaging triggers, respectively. Keep the following information about messaging connection aliases in mind:

- When you enable a Universal Messaging connection alias, the change takes effect immediately. Publishing services and triggers begin using the alias to send and receive messages from the messaging provider.
- When you enable a Broker connection alias, you must restart Integration Server before the change takes effect. Publishing services and webMethods messaging triggers do not begin using the alias to send and receive documents until after Integration Server restarts.

- The IS_LOCAL_CONNECTION messaging connection alias cannot be enabled or disabled. The alias is always available to Integration Server for local publishing.

➤ **To enable a messaging connection alias**

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. In the **webMethods Connection Alias Definitions** list, locate the messaging connection alias that you want to enable.
5. Click **No** in the **Enabled** column to enable the messaging connection alias.

For a Universal Messaging connection alias, Integration Server Administrator replaces **No** with ✓ **Yes**.

For a Broker connection alias, Integration Server Administrator replaces **No** with ✓ **Yes [Pending Restart]**.

6. If this is a Broker connection alias, restart Integration Server for your changes to take effect.

About Disabling a Message Connection Alias

You can disable any messaging connection alias on Integration Server. A disabled messaging connection alias cannot be used to send messages to or receive messages from the messaging provider specified in the alias. Specifically, when a messaging connection alias is disabled:

- Publishing services that use the disabled alias end with an `ISRuntimeException`.
- `webMethods` messaging triggers that use the alias are suspended.

When you disable a Universal Messaging connection alias, you do not need to restart Integration Server for the change to take effect.

- Publishing services that use the alias receive an `ISRuntimeException` as soon as the messaging connection alias is disabled.
- Before suspending `webMethods` messaging triggers that use the alias Integration Server waits a short time for triggers to finish processing messages the triggers already received. Integration Server considers message processing to be complete if the trigger service executes to completion (success or failure) and the trigger acknowledges the message to the message provider. If the trigger service is long running and cannot complete in the allotted time, Integration Server disables the trigger and considers message processing to be incomplete.

Note:

Suspending a trigger does not stop the trigger service. Consequently, even after Integration Server suspends the webMethods messaging trigger, the trigger service will execute to completion. After the trigger service completes, Integration Server attempts to acknowledge the message to the messaging provider. However, the acknowledgment attempt fails because the messaging connection alias is disabled. If the message is guaranteed, the messaging provider redelivers the message when the messaging connection alias is enabled and the trigger resumes. Redelivery of messages can result in duplicate processing.

When you disable a Broker connection alias, you must restart Integration Server for the change to the alias to take effect. The Broker connection alias is not actually disabled until Integration Server restarts. This means that publishing services and webMethods messaging triggers that use the alias continue to send and receive messages until restart occurs. Upon restart, Integration Server suspends any webMethods messaging trigger that use the messaging connection alias. Additionally, after restart, any publishing services that use the messaging connection alias will end with an `ISRuntimeException`.

Note:

The `IS_LOCAL_CONNECTION` messaging connection alias cannot be enabled or disabled. The alias is always available to Integration Server for local publishing.

Disabling a Messaging Connection Alias

You might disable a Universal Messaging connection alias to:

- Edit the messaging connection alias.
- Prevent services from publishing messages to a messaging provider.
- Stop receiving messages from a particular messaging provider.
- Stop all webMethods messaging triggers that use a particular messaging provider.

Because a Universal Messaging connection alias does not require an Integration Server restart for the change to take effect, disabling the alias is a quick way to stop the sending, retrieval, and processing of messages from the Universal Messaging server in the alias.

➤ To disable a messaging connection alias

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. In the **webMethods Connection Alias Definitions** list, locate the messaging connection alias that you want to disable.
5. Click the ✓ icon in the **Enabled** column to disable the messaging connection alias.

For a Universal Messaging connection alias, Integration Server Administrator replaces ✓ with **No**.

For a Broker connection alias, Integration Server Administrator replaces ✓ **Yes** with ✓ **No [Pending Restart]**.

- If this is a Broker connection alias, restart Integration Server for your changes to take effect.

Messaging Connection Alias Status

A messaging connection alias can have one of the following statuses in the **Enabled** column.

Status in Enabled Column	Description
✓ Yes	The messaging connection alias is enabled. Integration Server is connected to the messaging provider and can use the alias to send and receive messages.
✓ Yes [Pending Restart]	The messaging connection alias is currently disabled but will be enabled after Integration Server restarts. Integration Server is not connected to the messaging provider. Integration Server cannot use the alias to send and receive messages.
✓ Yes [Not Connected]	The messaging connection alias is enabled, but Integration Server is not connected to the messaging provider. Integration Server cannot use the alias to send and receive messages.
No	The messaging connection alias is not enabled. Integration Server cannot use the alias to send and receive messages.
✓ No [Pending Restart]	The messaging connection alias is currently enabled but will be disabled after Integration Server restarts. Currently, Integration Server is connected to Broker; services can publish messages to the Broker and triggers can retrieve messages from Broker.

Note: Integration Server Administrator does not display a status for the predefined IS_LOCAL_CONNECTION messaging connection alias. The alias cannot be enabled or disabled and is always available to Integration Server for local publishing.

Specifying the Default Messaging Connection Alias

The default messaging connection alias determines the messaging provider to which messages are sent when the **Connection alias name** property for the publishable document type is set to DEFAULT or is blank.

- Services that publish instances of the publishable document type use the default messaging connection alias to connect to the messaging provider and publish the document.
- webMethods messaging triggers that subscribe to that publishable document type use the default messaging connection alias to connect to the provider and retrieve the messages.

That is, when the publishable document type has a **Connection alias name** property set to DEFAULT or is blank, the default messaging connection alias determines which messaging provider receives and routes published instances of the document.

The first time Integration Server starts, Integration Server sets a default messaging connection alias based on whether Integration Server was migrated from a previous version and on what other products are installed in the same location.

➤ To change the default messaging connection alias

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. Click **Change Default Connection Alias**.
5. Under **Select New Default Connection Alias**, in the **Connection Alias Name** list, select the name of the messaging connection alias that you want to use as the new default messaging connection alias.
6. Click **Update**.

Notes

- After you change the default connection alias, Integration Server reloads all of the webMethods messaging triggers that subscribe to publishable document types that use the default messaging connection alias.
- After you change the default connection alias, you need to synchronize publishable document types that use the default alias with the messaging provider. This ensures that the provider definitions on the new default messaging provider are synchronized with publishable document types that use the default messaging connection alias.

Deleting a Messaging Connection Alias

Before you delete a messaging connection alias, keep the following in mind:

- The messaging connection alias must be disabled before it can be deleted. For more information about disabling a messaging connection alias, see [“Disabling a Messaging Connection Alias” on page 253](#).

- The messaging connection alias must not be assigned to a publishable document type when you delete it. Integration Server does not prevent you from deleting a messaging connection alias used by a publishable document type.
- The predefined messaging connection alias IS_LOCAL_CONNECTION cannot be deleted.
- The predefined messaging connection alias IS_DES_CONNECTION cannot be deleted.

➤ **To delete a messaging connection alias**

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**
3. Under **webMethods Messaging Configuration**, click **webMethods Messaging Settings**.
4. In the **webMethods Connection Alias Definitions** table, disable the messaging connection alias if it is not yet disabled.
5. In the **Delete** column, click **×** for the messaging connection alias that you want to delete.0
6. Click **OK** to verify that you want to delete the messaging connection alias.

Authenticating Connections to the Universal Messaging Server

When Software AG Universal Messaging serves as the messaging provider, Integration Server acts as the client, and the Universal Messaging realm server acts as the server. Communication between Integration Server and the Universal Messaging server is secured by way of ACL management on the Universal Messaging server. You can add an extra layer of security for a connection between Integration Server and Universal Messaging, before the Universal Messaging server applies the ACLs, by authenticating the connection with a user name and password. You specify this authentication method when you create the Universal Messaging connection alias.

Note:

This feature applies to Integration Server 9.8 or later and Universal Messaging 9.6 or later.

When you configure a Universal Messaging connection alias to authenticate connections between Integration Server and Universal Messaging in this way, user credentials are exchanged using either the Simple Authentication and Security Layer (SASL) framework or the Java Authentication and Authorization Service (JAAS) framework. Universal Messaging uses the JAAS framework by default.

The Universal Messaging administrator determines which framework to use. Work with the Universal Messaging administrator to have the following items in place before you test the connection:

- **SASL framework:** With this framework, the Universal Messaging server verifies user credentials based on a specified Directory instance, and credentials are stored in an internal directory or

in an external directory such as LDAP. Work with the Universal Messaging administrator to have the following items in place:

- **Internal user repository.** If you want to store user credentials in an internal user repository, the Universal Messaging administrator should create the repository using the Software AG Security Infrastructure command line tool. The Universal Messaging administrator should also set the `Nirvana.directory.provider` system property as indicated in the server property configuration table that follows.
- **External user repository configuration.** If you want to store user credentials in an external repository such as LDAP, the Universal Messaging administrator should set the `Nirvana.directory.provider` system property as indicated in the server property configuration table that follows.
- **Server property configuration.** The Universal Messaging administrator should set properties in the `nserver.conf` file, located in `Universal Messaging_directory \server \umserver \bin \` on the Universal Messaging server, as follows:

Set this property	To
<code>Nirvana.auth.sagrepo.path</code>	The relative or absolute path of the user credentials text file that the Security Infrastructure command line tool created
<code>Nirvana.directory.provider</code>	One of the following: <ul style="list-style-type: none"> ■ If user credentials are stored in an internal user repository, set this property to: <code>com.pcbsys.foundation.security.auth.fSAGInternalUserRepositoryA</code> ■ If user credentials are stored in an external user repository, set this property to: <code>com.pcbsys.foundation.security.auth.fLDAPAdapter</code>
<code>Nirvana.auth.enabled</code>	Y If this parameter is set to N and credentials are passed with the connection request, the Universal Messaging server ignores the credentials and connects without authentication.
<code>Nirvana.auth.server.mandatory</code>	Y

For more information about internal and external user repositories, the Security Infrastructure command line tool, and the `nserver.cnf` system properties, see the *Software AG Infrastructure Administrator's Guide* and the Universal Messaging documentation.

- **JAAS framework:** With this framework, the Universal Messaging server invokes a JAAS login module to verify user credentials. Organizations might choose this method over SASL when they require custom logic for authenticating users (for example, when a custom service is

needed to extract user credentials from an external database). Work with the Universal Messaging administrator to have the following items in place:

- **Login module.** The login module contains the code that retrieves and validates passwords. If the user name/password combination supplied in the connection alias differs from what is specified in the login module, Integration Server displays an error message indicating that the connection failed.
- **JAAS configuration file.** The Universal Messaging administrator should create a JAAS configuration file, which defines the JAAS key and calls the login module, on the Universal Messaging server.
- **Server property configuration.** The Universal Messaging administrator should set properties in the `nserver.conf` file, located in `Universal Messaging_directory \server\umserver\bin\` on the Universal Messaging server, as follows:

Set this property	To
<code>Nirvana.auth.server.jaaskey</code>	<code>noauth</code>
<code>wrapper.java.classpath</code>	The login module location

For more information about login modules, JAAS configuration files, and the `nserver.cnf` system properties, see the *Software AG Infrastructure Administrator's Guide* and the Universal Messaging documentation.

Specifying the Keep-Alive Mode for the Broker Connection

After configuring the connection to the Broker, you can specify the keep-alive mode that you want Integration Server to use.

The *keep-alive mode* indicates whether the Broker checks for dropped connections from a client and then explicitly disconnects the client if it has dropped the connection. By disconnecting the client, the Broker makes any unacknowledged documents retrieved by that client available for redelivery to other clients.

Note:

You can specify a keep-alive mode only if Integration Server connects to a webMethods Broker version 6.1 or later.

If client state is not shared, an undetected broken connection does not pose a problem. The Broker will automatically redeliver unacknowledged events to the client when it reconnects. However, if the client state is shared and a client loses its connection to the Broker, the client cannot retrieve the unacknowledged documents after it re-establishes the connection. (The default client for the Integration Server and all trigger clients are shared-state clients.) This is because the same client ID is used by all the clients in a shared-state client. The Broker cannot distinguish the reconnection of one client from the ordinary reconnections of other clients with the same client ID. The unacknowledged documents retrieved by the now disconnected client will not be made available for redelivery until the Broker receives an explicit disconnect notice (generally, when the TCP/IP connection finally times out). In some cases, this might be hours later.

To avoid a situation in which unacknowledged documents stay on the Broker for an unacceptable period of time, you can select a keep-alive mode that will disconnect unresponsive clients and make unacknowledged documents available for redelivery.

Note:

For more information about the Broker keep-alive feature and about shared-state clients, see the *webMethods Broker Java Client API Reference*.

You can configure one of the following keep-alive modes:

- **Normal.** The Broker sends a keep-alive message to the Integration Server at a specified time interval (keep-alive period) and expects a response within another specified time interval (max response time). If the Broker does not receive a response, it will retry up to the number of times specified by the retry count. If the Integration Server still does not respond to the keep-alive message, the Broker explicitly disconnects the Integration Server. Normal is the default mode.

For example, by default, the Broker sends the Integration Server a keep-alive message every 60 seconds. If the Integration Server does not respond within 60 seconds, the Broker sends up to three more keep-alive messages before disconnecting the Integration Server. (The default retry count is 3.)

- **Listen Only.** The Broker disconnects the Integration Server if there is no activity from the Integration Server over a specified time interval (keep-alive period). In listen only mode, the Broker does not send keep-alive messages to the Integration Server and ignores the retry count.

For example, suppose that the Broker expects activity from the Integration Server every 60 seconds. If there is no activity from the Broker after 60 seconds, the Broker disconnects the Integration Server.

- **Disabled.** The Broker disables keep-alive interaction with this Integration Server. The Broker does not send keep-alive messages and does not disconnect the Integration Server because of inactivity.

Note:

The Broker does not communicate directly with Integration Server. The Broker Client API facilitates communication between Broker and Integration Server.

Setting Server Configuration Parameters for Keep-Alive Mode

The keep-alive mode is determined by the combination of values for the following set of server configuration parameters:

- **watt.server.brokerTransport.dur.** Specifies the number of seconds of idle time that the Broker waits before sending a keep-alive message to Integration Server. This is the keep-alive period. The default is 60 seconds.
- **watt.server.brokerTransport.max.** Specifies the number of seconds that the Broker waits for the Integration Server to respond to a keep-alive message. This is the max response time. The default is 60 seconds.

- **watt.server.brokerTransport.ret.** Specifies the number of times the Broker re-sends keep-alive messages before disconnecting an un-responsive Integration Server. This is the retry count. The default is 3.

For information about setting a keep-alive mode using these parameters, see the sections that follow. For more information about these parameters, see [“watt.server.” on page 1053](#).

Normal Mode

Use the settings in the following table to configure normal keep-alive mode. This is the default mode.

Parameter	Value
watt.server.brokerTransport.dur	Any integer greater than 0 but less than 2147483647. The default is 60.
watt.server.brokerTransport.max	Any integer greater than 0 but less than or equal to 2147483647. The default is 60.
watt.server.brokerTransport.ret	Any integer between 1 and 2147483647. The default is 3.

Listen Only Mode

Use the settings in the following table to configure listen only keep-alive mode.

Parameter...	Value
watt.server.brokerTransport.dur	2147483647
watt.server.brokerTransport.max	Any integer greater than zero but less than 2147483647
watt.server.brokerTransport.ret	N/A. The retry count is ignored in listen only mode.

Disabled

Use the settings in the following table to disable keep-alive mode.

Parameter	Value
watt.server.brokerTransport.dur	2147483647
watt.server.brokerTransport.max	2147483647
watt.server.brokerTransport.ret	1

Synchronizing Broker Clients When the Primary Port for Integration Server Changes

It is important to establish the primary port for your Integration Server *before* connecting your server to the Broker. However, if you change the Integration Server's primary port number after configuring a Broker connection alias, the Broker client for your Integration Server may become unsynchronized with your Broker's configuration.

If you change the Integration Server's primary port after configuring the Broker connection alias, you need to synchronize your Broker clients to the Integration Server's new port configuration.

➤ To synchronize Broker clients with the Integration Server's primary port

1. Using the Broker user interface, delete the clients that reflect the server's original primary port number, for example 10.3.33.129_5555_DefaultClient.
2. Delete the `dispatch.cnf` file from the `Integration Server_directory \instances \instance_name \config` directory.
3. Reconfigure the Broker connection alias, using the procedure described in [“Creating a Broker Connection Alias” on page 234](#).

Configuring Document Stores

Integration Server uses *document stores* to save documents to disk or to memory while the documents are in transit or waiting to be processed. Integration Server maintains three document stores for published documents.

- **Default document store.** The default document store contains guaranteed documents delivered to the default Broker client for the Integration Server. When Integration Server retrieves documents delivered to its default Broker client, Integration Server places the documents in the default document store. Documents remain in the default document store until the dispatcher determines which triggers subscribe to the document. The dispatcher then moves the documents to the trigger queues for the subscribing triggers.
- **Trigger document store.** The trigger document store contains locally published guaranteed documents that are delivered to a specific webMethods messaging triggers. For each trigger, Integration Server creates a queue for locally published guaranteed documents in the trigger document store. A guaranteed document remains in the trigger queue within the trigger document store until the server successfully processes the document.
- **Outbound document store.** The outbound document store, sometimes called a client-side queue, contains documents waiting to be sent to the Broker. Integration Server places documents in the outbound document store when the configured Broker is not available. When the connection to the Broker is restored, Integration Server empties the outbound document store by sending the saved documents to the Broker.

Note:

The outbound document store is used for guaranteed documents published to the Broker only.

Using Integration Server Administrator, you can configure properties for each document store.

Configuring the Default Document Store

The default document store contains published documents delivered directly to the default Broker client for an Integration Server. Documents remain in the default document store until the dispatcher moves the document to the trigger queues for the subscribing webMethods messaging triggers.

Note:

You do not need to configure the default document store if you use Universal Messaging as the messaging provider.

> To configure the default document store

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Resources**.
3. Click **Store Settings**, and then click **Edit Document Store Settings**.
4. Set the **Default Document Store** fields as follows:

Field	Description
Store Location	<p>The location of the default document store. By default, the Integration Server saves document stores in the following directory:</p> <p><i>Integration Server_directory \instances \instance_name \DocumentStore</i></p> <p>If you want to save the default document store in a different location, specify the directory in this field. If the directory does not exist, the server creates it.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: Make sure that you have write access to the specified directory and that the directory does not contain any characters considered illegal by your operating system.</p> </div>
Initial Store Size (MB)	<p>The amount of disk space allocated to the default document store at start up. The default store automatically increases when it receives data that exceeds the initial size. The default size is 25MB.</p> <p>When setting the Initial Store Size, consider the size and volume of documents that you expect to be delivered to the default document</p>

Field	Description
	<p>store. If you expect large documents or a high volume of documents, consider increasing the Initial Store Size.</p> <p>Important: Make sure that there is enough free disk space on the Integration Server machine to accommodate the initial sizes of the default document store, the trigger document store, and the XA recovery store.</p>
Capacity	<p>The maximum number of documents in the default document store. The default is 10 documents.</p> <p>The Capacity must be greater than the Refill Level.</p> <p>If you set Capacity to 0, the server automatically suspends the Refill Level. If you set the Capacity field to 0, the server displays "Suspended" next to the field on the Settings > Resources > Store Settings page.</p> <p>Note: The Capacity field displays " Broker Not Configured" if there is not a Broker configured for the server.</p>
Refill Level	<p>The number of unprocessed documents that remain in the default document store before the Integration Server retrieves more documents from the Broker.</p> <p>For example, if you assign the default document store a Capacity of 10 and a Refill Level of 4, the server initially retrieves ten documents. When only four documents remain to be processed in the default document store, the server retrieves six more documents. If six documents are not available, the server retrieves as many as possible.</p> <p>The default refill level is 4 documents.</p> <p>The Refill Level must be less than Capacity. If you set Capacity to 0, the server automatically suspends the Refill Level.</p> <p>Note: The Refill Level field displays " Broker Not Configured" if there is not a Broker configured for the server.</p>

5. Click **Save Changes**.
6. If you changed one or more parameters that require server restart for the new value to take effect, restart Integration Server.

Note:

An asterisk (*) next to a parameter indicates that you need to restart the server for changes to take effect.

About the Trigger Document Store

The trigger document store contains trigger queues in which Integration Server keeps locally published guaranteed documents awaiting processing. A document remains in trigger queue in the trigger document store until one of the following occurs:

- Integration Server successfully executes the trigger service specified in the trigger condition satisfied by the document.
- Integration Server discards the document because the document does not satisfy any conditions in the trigger.
- Integration Server discards the document because it is a duplicate of one already processed by the trigger. This can occur only if the trigger is configured for exactly-once processing.
- Integration Server cannot determine whether the trigger processed the document previously, assigns the document a status of In Doubt, and instructs the audit subsystem to log the document. This can occur only if the trigger is configured for exactly-once processing.

Configuring the Trigger Document Store

When you configure the trigger document store, you specify the location of the store and the initial size of the store.

Note:

You do not need to configure the trigger document store if you use Universal Messaging as the messaging provider.

> To configure the trigger document store

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Resources**.
3. Click **Store Settings**, and then click **Edit Document Store Settings**.
4. Set the **Trigger Document Store** fields as follows:

Field	Description
Store Location	The location of the trigger document store. By default, Integration Server saves the trigger document store in the following directory: <i>Integration Server_directory \instances \instance_name \DocumentStore</i>

Field	Description
	<p>If you want to save the trigger document store in a different directory, specify the directory in this field. If the directory does not exist, the server creates it.</p> <p>Important: Make sure that you have write access to the specified directory and that the directory does not contain any characters considered illegal by your operating system.</p>
<p>Initial Store Size (MB)</p>	<p>The amount of disk space allocated to the trigger document store at start up. The trigger document store automatically increases when it receives data that exceeds the initial size. The default size is 35MB.</p> <p>Important: Make sure that there is enough free disk space on the Integration Server machine to accommodate the initial sizes of the default document store, the trigger document store, and the XA recovery store.</p>

5. Click **Save Changes**.
6. If you changed one or more parameters that require server restart for the new value to take effect, restart Integration Server.

Note:
An asterisk (*) next to a parameter indicates that you need to restart the server for changes to take effect.

Maintaining Inbound Document History for Received Documents

If Integration Server connects to a Broker version 6.0.1, you can configure the **Inbound Document History** setting to maintain a history of documents received by the server. This instructs the server to perform a very basic form of duplicate detection for all triggers.

If Integration Server connects to a Broker version 6.1 or later, you can configure duplicate detection on a per trigger basis. For information about configuring duplicate detection for webMethods messaging triggers using version 6.1 or later of Integration Server, see the *webMethods Service Development Help*.

In a cluster of Integration Servers connected to a Broker version 6.0.1, each Integration Server in the cluster maintains its own inbound document history information. That is, the inbound document history information is not shared across the cluster.

Note:
The **Inbound Document History (minutes)** field can be set only if Integration Server connects to a Broker version 6.0.1. The field is not available if the server connects to a 6.1 or later version

of the Broker. For detailed information about configuring inbound document history, see the *webMethods Integration Server Administrator's Guide* version 6.0.1.

Enabling Inbound Client-Side Queuing

If Integration Server connects to a 6.0.1 version of the Broker, you can use inbound client-side queuing. When inbound client-side queuing is enabled, Integration Server stores received documents on disk and acknowledges documents to the Broker immediately after receipt and storage. When inbound client-side queuing is disabled, Integration Server stores received documents in memory and acknowledges documents to the webMethods Broker after processing completes.

Note:

Inbound client-side queuing is not available when Integration Server connects to a 6.1 or later version of the Broker. For information about using client-side queuing with a 6.0.1 version of the Broker, see the *webMethods Integration Server Administrator's Guide* version 6.0.1.

About the Outbound Document Store

The outbound document store, sometimes called a client side queue (CSQ), contains guaranteed documents published by Integration Server when the Broker specified in the Broker connection alias is not available. After the connection to the Broker is re-established, Integration Server sends the documents in the outbound document store to the Broker.

Note:

The outbound document store is not used for guaranteed documents published to Universal Messaging. Instead, each Universal Messaging connection alias has its own client side queue. Configuring the client side queue for a Universal Messaging is part of creating the Universal Messaging connection alias. For more information about creating a Universal Messaging connection alias, see [“Creating a Universal Messaging Connection Alias” on page 238](#).

For Integration Server to use the outbound document store, the `watt.server.publish.useCSQ` parameter must be set to “always”. If `watt.server.publish.useCSQ` parameter is set to “never”, then Integration Server throws a `ServiceException` when the Broker is unavailable at the time documents are published to the Broker.

If the initial attempt to publish the document to Broker from the CSQ fails, Integration Server makes subsequent attempts until the document is published successfully or Integration Server makes the maximum attempts specified in `watt.server.publish.maxCSQRedeliveryCount`. Each attempt to publish to Broker from the CSQ is considered a redelivery attempt. After Integration Server makes the specified number of attempts to transmit a document from the CSQ to the Broker and all attempts fail, the audit subsystem logs the document and assigns it a status of `STATUS_TOO_MANY_TRIES`.

Integration Server can empty documents from the outbound document store in order of publication or in parallel. The value of the `watt.server.publish.drainCSQInOrder` parameter determines how the outbound store is emptied. By default, Integration Server sends all newly published documents (guaranteed and volatile) to the outbound document store until the outbound store has been emptied. This allows Integration Server to maintain publication order. When Integration Server

is configured to empty the outbound store in parallel, the outbound store is emptied while new documents are being published to the Broker.

Configuring the Rate at which Integration Server Drains the Outbound Document Store

You can configure how quickly the server empties the outbound document store by setting the **Maximum Documents to Send per Transaction** parameter on the **Settings > Resources > Store Settings > Edit Document Store Settings** page. By default, this parameter is set to 25 documents. To empty the outbound document store more quickly, increase the number of documents sent per transaction. Keep in mind that the amount of memory needed to send documents increases with the number of documents sent for each transaction. If you want to use less memory to empty the outbound document store and can allow the outbound document store to empty more slowly, decrease the number of documents sent for each transaction. However, it is advisable to drain the outbound document store as quickly as possible because Integration Server performs more quickly and uses fewer resources when publishing documents directly to the Broker.

You only need to configure the rate at which Integration Server drains the outbound document store (client side queue) if:

- Integration Server publishes documents to the Broker. Integration Server does not use the outbound document store when publishing documents to Universal Messaging.
- The outbound document store is enabled for Integration Server. That is, the `watt.server.publish.useCSQ` parameter is set to "always".

Tip:

You can monitor the number of documents in the outbound document store by checking the value of the **CSQ Count** field for the Broker connection alias on the **Settings > Messaging > webMethods Messaging Settings** screen.

➤ To configure how quickly Integration Server empties the outbound document store

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Resources**.
3. Click **Store Settings**, and then click **Edit Document Store Settings**.
4. Under **Outbound Document Store**, in the **Maximum Documents to Send per Transaction**, type the number of documents the server should send from the outbound document store to the Broker for each transaction.

If there is no configured Broker, the Integration Server Administrator displays " Broker Not Configured" next to the field name.

5. Click **Save Changes**.

Setting the Capacity of the Outbound Document Store

By default, the outbound document store (client side queue) can contain a maximum of 500,000 documents. After the outbound document store reaches capacity, the server "blocks" or "pauses" any threads that are executing services that publish documents. The threads remain blocked until the server begins draining the outbound document store.

The `watt.server.control.maxPersist` server parameter determines the capacity of the outbound document store. If you plan to bring the Broker down for an extended time period, consider editing this parameter to lower the capacity of the outbound document store. If you keep the outbound document store at the default capacity, and the Broker becomes unavailable, it is possible that storing outbound documents could exhaust memory and cause Integration Server to fail. If the outbound document store has a lower capacity, Integration Server will block threads instead of continuing to use memory by storing documents.

Associating a User Account with webMethods Messaging Trigger Services

When a client invokes a service via an HTTP request, Integration Server checks the credentials and user group membership of the client against the Execute ACL assigned to the service. Integration Server performs this check to make sure the client is allowed to invoke that service. In a publish-and-subscribe situation, however, Integration Server invokes the service when it receives a document rather than as a result of a client request. Because Integration Server does not associate user credentials with a published document, you can specify the user account for Integration Server to use when invoking services associated with webMethods messaging triggers.

You can instruct Integration Server to invoke a service using the credentials of one of the predefined user accounts (Administrator, Central, Default, Developer, Replicator). You can also specify a user account that you or another server administrator defined. When Integration Server receives a document that satisfies a trigger condition, Integration Server uses the credentials for the specified user account to invoke the service specified in the trigger condition.

Make sure that the user account you select includes the credentials required by the execute ACLs assigned to the services associated with triggers. For example, suppose that you specify "Developer" as the user account for invoking services in triggers. The `receiveCustomerInfo` trigger contains a condition that associates a publishable document type with the service `addCustomer`. The `addCustomer` service specifies "Replicator" for the Execute ACL. When the trigger condition is met, the `addCustomer` service will not execute because the user setting you selected (Developer) does not have the necessary credentials to invoke the service (Replicator).

How you specify a user account for a webMethods messaging trigger service depends on the messaging provider used by the trigger.

- If the trigger receives message from Broker, specify the user account using Integration Server Administrator. The user account you specify applies to all webMethods messaging triggers that receive messages from the Broker. For more information, see [“Specifying a User Account for Invoking webMethods Messaging Trigger Services”](#) on page 269.

- If the trigger receives messages from Universal Messaging, specify the user account for a trigger using Designer. For each webMethods messaging trigger that retrieves messages from Universal Messaging, you use the **Execution user** property to indicate which users can invoke the trigger services for that trigger. For more information about setting the execution user for a trigger, see *webMethods Service Development Help*.

Specifying a User Account for Invoking webMethods Messaging Trigger Services

For webMethods messaging triggers that receive messages from Broker, you set the execution user using Broker. The user account you specify applies to all webMethods messaging triggers that receive messages from the Broker.

➤ To specify a user account to execute a webMethods messaging trigger service

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Resources**.
3. Click **Store Settings**, and then click **Edit Document Store Settings**.
4. In the **Trigger Document Store** area of the screen, in the **User** field, select the user account whose credentials the Integration Server uses to execute a service specified in a trigger condition. The user account can be selected from a central or external directory. The default is Administrator.
5. Click **Save Changes**.

Load Balancing with a Non-Clustered Group of Integration Servers

Integration Servers can receive messages from the same messaging provider in a load-balanced manner if the Integration Servers are connected to the messaging provider as a *non-clustered group*. In a non-clustered group, multiple Integration Servers act as a single messaging client but are not members of a cluster.

Note:

In addition to the term “non-clustered group,” the terms “stateless cluster” and “external cluster” are sometimes used to describe the situation in which a group of Integration Servers function in a manner similar to a cluster but are not part of a configured cluster.

Integration Servers can receive messages from the same messaging provider in a load-balanced manner if the Integration Servers are connected to the messaging provider as a *non-clustered group*.

To receive messages in a load-balanced fashion, each Integration Server must:

- **Connect to the same messaging provider.** A group of non-clustered Integration Servers can receive messages in a load-balanced fashion from Universal Messaging or Broker. The messaging connection aliases should be the same on each Integration Server.
- **Use the same client prefix.** Use the Integration Server Administrator to specify the client prefix for your Integration Servers. For instructions, see [“Creating a Broker Connection Alias” on page 234](#) and [“Creating a Universal Messaging Connection Alias” on page 238](#).
- **Use the same client group.** When Broker is the messaging provider, each Integration Server must use the same client group. Use the Integration Server Administrator to specify the client group for your Integration Servers. For instructions, see [“Creating a Broker Connection Alias” on page 234](#).
- **Have the same triggers.** To receive message in a load-balanced manner, the non-clustered Integration Servers must act as a single messaging client, which includes having the same document subscriptions. For information about configuring webMethods messaging triggers, see *webMethods Service Development Help*.

Integration Servers in a non-clustered group can also share a document history database and a cross-reference database without being in an Integration Server cluster.

Note:

A stateless cluster of Integration Servers does not use a Terracotta Server Array.

Important Considerations for Using a Stateless Cluster with webMethods Messaging

Keep the following points in mind when determining whether to connect a non-clustered group of Integration Servers to the same messaging provider for load-balancing:

- The Integration Servers in the group must be the same version and at the same fix level.
- The Integration Servers in the group can share a document history database and a cross-reference database.
- The actions that Integration Server takes when modifying triggers in a non-clustered group Integration Servers depends on the value of the **Client Prefix Is Shared** option:
 - If the **Client Prefix Is Shared** option for the Integration Server connection to the messaging provider is set to **Yes**, Integration Server prevents updates to shared objects on the messaging provider.

In a non-clustered group of Integration Servers for which the **Client Prefix Is Shared** option is **Yes**, it is your responsibility to manually update the settings on the messaging provider.

- If the **Client Prefix Is Shared** option for the Integration Server connection to the messaging provider is set to **No**, Integration Server allows updates to shared objects, such as a queue or channel, on the messaging provider. Modifying a webMethods messaging trigger on one of the Integration Servers could delete documents and other data on the messaging provider. For example, when the messaging provider is Broker, if you enable a trigger,

disable a trigger, or change the processing mode, the Integration Server deletes and then recreates the associated trigger client queue on the Broker. This action deletes any documents that are in the trigger client queue on the Broker.

12 Configuring Integration Server for JMS

Messaging

■ Overview of JMS Messaging Configuration	274
■ Working with JNDI Providers	274
■ Working with JMS Connection Aliases	284
■ Creating Administered Objects	307
■ Monitoring a Connection Factory Object for Changes	309
■ Using SSL with JMS	313
■ Supported JMS Providers	314
■ Adding JMS Provider Client Libraries to Integration Server Classpath	319

12.1 Overview of JMS Messaging Configuration

To configure Integration Server for JMS messaging, you need to:

- Create one or more JNDI provider aliases to specify where Integration Server can look up administered objects when it needs create a connection to JMS provider or specify a destination for sending or receiving messages.
- Create one or more connection aliases that encapsulate the properties that Integration Server needs to create a connection with the JMS provider.

Integration Server also includes various server configuration properties that you can use to change the default server behavior. For a list of server configuration parameters related to JMS, see [“Server Configuration Parameters” on page 1017](#).

Working with JNDI Providers

Each JMS provider can store JMS administered objects in a standardize namespace called the Java Naming and Directory Interface (JNDI). JNDI is a Java API that provides naming and directory functionality to Java applications.

As the JMS client, Integration Server uses a JNDI provider alias to encapsulate the information needed to look up an administered object. When you create a JMS connection alias, you can specify the JNDI provider alias that Integration Server should use to look up administered objects (i.e., Connection Factories and Destinations).

Note:

If you connect directly to webMethods Broker using the native webMethods API, you do not need to use a JNDI provider. If you choose to use webMethods Broker but do not want to connect natively (using the native webMethods API), you still need to use a JNDI provider. For more information about using the native webMethods API to connect to webMethods Broker, see [“Connecting to webMethods Broker with the Native webMethods API” on page 284](#).

Predefined JNDI Provider Aliases

Integration Server includes predefined JNDI provider aliases. Integration Server creates the aliases the first time Integration Server starts. If the aliases existed in an Integration Server that was migrated from a previous version, Integration Server keeps the existing aliases. That is, Integration Server does not overwrite the previous alias definition.

The predefined JNDI provider alias for Integration Server is `DEFAULT_IS_JNDI_PROVIDER`, a JNDI provider alias with predefined settings for establishing a connection to the local instance of Software AG Universal Messaging. If Software AG Universal Messaging is not installed in the same directory as Integration Server, then the `DEFAULT_IS_JNDI_PROVIDER` uses a provider URL of `nsp://localhost:9000`.

Note:

If you migrated to Integration Server version 9.10 or later from an earlier version, then you might have the `EventBusJndiProvider` alias, which was a predefined JNDI provider alias in versions prior to 9.10. This alias points to the local instance of Software AG Universal Messaging. If Software AG Universal Messaging is not installed in the same directory as Integration Server, then the `EventBusJndiProvider` uses a provider URL of `nsp://localhost:9000`.

Creating a JNDI Provider Alias

A JNDI provider alias contains the information needed to look up an administered object. Most JMS connection aliases require a JNDI provider alias to create connections and look up destinations.

Keep the following information in mind when you create a JNDI provider alias:

- If the JMS provider requires the use of SSL for sending and receiving messages, you might be able to include SSL certificate information in the JNDI provider alias.

For more information about including SSL certificate information in the JNDI provider alias, see [“Including SSL Configuration Information in the JNDI Provider Alias” on page 281](#). For information about configuring SSL communication with the JMS provider, see [“Using SSL with JMS” on page 313](#).

- If you connect directly to webMethods Broker using the native webMethods API, you do not need to use a JNDI provider and, therefore, do not need to create a JNDI provider alias.
- If you choose to use webMethods Broker but do not want to connect natively (using the native webMethods API), you still need to use a JNDI provider. For more information about using the native webMethods API to connect to webMethods Broker, see [“Connecting to webMethods Broker with the Native webMethods API” on page 284](#).

Use the following procedure to create an alias to a JNDI provider.

➤ To create a JNDI provider alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JNDI Settings**.
4. Click **Create JNDI Provider Alias**.
5. Specify the following information for the JNDI provider alias:

Field	Description
JNDI Alias Name	The alias name that you want to assign to this JNDI provider.
Description	A description for this JNDI alias.

Field	Description
Predefined JNDI Templates	<p>The JNDI template that you want to use.</p> <p>The JNDI templates provide information that you can use to complete alias configuration for a specific provider.</p> <p>Note: After you create a JNDI provider, Integration Server Administrator displays Current Settings as the value of the Predefined JNDI Templates field. This indicates that Integration Server uses the currently specified settings for the JNDI provider alias.</p>
Initial Context Factory	<p>The class name of the JNDI provider. The JNDI provider uses the initial context as the starting point for resolving names for naming and directory operations.</p> <p>If you selected a predefined JNDI template, Integration Server displays the initial context factory for the provider.</p>
Provider URL	<p>The primary URL of the initial context for sessions with the JNDI provider. The URL specifies the JNDI directory in which the JNDI provider stores JMS administered objects.</p> <p>If you selected a predefined JNDI template, replace the placeholder information in brackets << >> with information specific to your configuration.</p> <p>If you are using Software AG Universal Messaging, this is the Universal Messaging realm server in the format <code>nsp://UM_host:UM_port</code> (for example, <code>nsp://127.0.0.1:9000</code>).</p> <p>If you are using a cluster of Universal Messaging realm servers, supply a list of the URLs to each realm server in the cluster. Use a colon or semi-colon to separate each URL:</p> <ul style="list-style-type: none"> ■ Separate the URLs using a comma if Integration Server always attempts to connect to the first Universal Messaging server in the list, only trying the second Universal Messaging server in the list if the first server becomes unavailable ■ Separate the URLs using a semicolon if Integration Server connects to a randomly chosen URL from the list. This may result in better distribution of clients across the servers in the cluster.
	<p>Note: You must specify a URL for <i>every</i> Universal Messaging server in the cluster. Integration Server connects only to the Universal Messaging servers for which you specify a URL. Consequently,</p>

Field	Description
	<p>Integration Server creates resources such as destinations or durable subscribers on the specified Universal Messaging servers. If you specify only one Universal Messaging server in a cluster, the resources will be local to that server only.</p> <p>Note: If the JMS connection alias uses this JNDI provider alias to connect to Universal Messaging cluster and the JMS connection alias is configured to follow the master, Integration Server always connects to the master realm server in the cluster. By default, Integration Server is configured to follow the master realm server in a cluster. Software AG recommends using a comma-separated list of URLs if the JMS connection alias follows the master. For more information about the Enable Follow the Master option for a JMS connection alias, see “Creating a JMS Connection Alias” on page 285</p>
Provider URL Failover List	<p>A list of the failover URLs of the initial context for sessions with the JNDI provider. Specify one URL per line. If the connection to the primary JNDI provider becomes unavailable, Integration Server automatically attempts a connection to a JNDI provider specified in this list.</p> <p>For more information about setting up a JNDI provider failover list, see “Creating a JNDI Provider Failover List” on page 280.</p>
Security Principal	<p>The principal name, or user name supplied by Integration Server to the JNDI provider, if the provider requires one for accessing the JNDI directory.</p> <p>For information about whether or not the JNDI provider requires security principal information, consult the product documentation for the JNDI provider.</p>
Security Credentials	<p>The credentials, or password, that Integration Server provides to the JNDI provider, if the provider requires security credentials to access the JNDI directory.</p> <p>For information about whether or not the JNDI provider requires security credentials, consult the product documentation for the JNDI provider.</p>
Other Properties	<p>Any additional properties the JNDI provider requires for configuration. For example, you might need to specify the classpath for any additional .jar or class files that the JNDI provider needs to connect to the JNDI.</p>

Field	Description
	<p>When you select a predefined JNDI template, Integration Server populates this field with any additional properties and placeholder information required by the JNDI provider.</p> <p>For more information about additional properties or classes required by a JNDI provider and the location of those files, see the product documentation for the JNDI provider.</p>

6. For **Use SSL**, select one of the following:

- Select **Yes** if you want to use SSL to connect to the JNDI provider and specify SSL certificate information in the JNDI provider alias. When it creates the JNDI context, Integration Server passes the certificate information into the JNDI context.

For more information, see [“Including SSL Configuration Information in the JNDI Provider Alias” on page 281](#).

Note:

This approach is only useful when the JNDI provider and JMS provider use the same set of certificates such as when using Universal Messaging as the JMS and JNDI providers.

- Select **No** if you do not want to include SSL configuration information in the JNDI provider alias. This is the default value.

If you select **No** and the JMS provider requires that communications be secured via SSL, you must configure Integration Server to provide certificate information a different way such as via the `javax.net.ssl` properties. For information about configuring SSL communication with the JMS provider, see [“Using SSL with JMS” on page 313](#).

7. If you selected **Yes** for **Use SSL**, under **SSL Settings**, identify the truststore and/or keystore that contains the certificates used to establish a SSL connection to the JNDI and JMS providers.

Field	Description
Truststore Alias	<p>The alias for the truststore that contains the certificates of the Certificate Authority (CA) for the JNDI and JMS providers.</p> <p>You must select a truststore alias if you want to set up SSL communication.</p>
Keystore Alias	<p>The alias for the keystore that contains the client certificates for Integration Server to use when connecting to the JNDI and JMS providers.</p> <p>You must select a keystore alias if you want to set up two-way SSL communication.</p>

Field	Description
Key Alias	The alias to the key that contains the private key for connecting to the JNDI and JMS providers. The key alias must be in the keystore specified in Keystore Alias .

Note:

If the JMS provider requires that the SSL settings of the JMS client JVM must be used to establish the SSL connection, the certificates in the truststore and keystores specified under **Secure Settings** are ignored.

- If you selected **Yes** for **Use SSL**, under **JNDI Property Names (JNDI Provider Specific)**, specify the JNDI property names that the JNDI provider uses to establish a secure connection. JNDI property names vary per JNDI provider.

Field	Description
Truststore Property Name	The name of the JNDI property for storing the truststore location. This property name is required if you selected a Truststore Alias in SSL Settings .
Truststore Password Property Name	The name of the JNDI property for storing the truststore password. This property name is required if you selected a Truststore Alias in SSL Settings .
Keystore Property Name	The name of the JNDI property for storing the keystore location. This property name is required if you selected a Keystore Alias in SSL Settings .
Keystore Password Property Name	The name of the JNDI property for storing the keystore password. This property name is required if you selected a Keystore Alias in SSL Settings .
Keystore Format Property Name	The name of the JNDI property for storing the keystore format. This property name is optional.
Private Key Property Name	The name of the JNDI property for storing the private key name. This property name is optional.

- Click **Save Changes**.

Editing a JNDI Provider Alias

Use the following procedure to edit an existing JNDI provider alias.

➤ To edit a JNDI provider alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JNDI Settings**.
4. In the JNDI Provider Alias Definitions list, select the JNDI provider alias that you want to edit. Integration Server Administrator displays details about the JNDI provider alias.
5. Click **Edit JNDI Provider Alias**.
6. Edit the properties of the JNDI provider alias.

For more information about the fields, see [“Creating a JNDI Provider Alias” on page 275](#).

7. Click **Save Changes**.

Deleting a JNDI Provider Alias

Use the following procedure to delete a JNDI provider alias.

Important:

When you delete a JNDI provider alias, any service or JMS trigger that uses a JMS connection alias that relies on the JNDI provider alias will fail.

➤ To delete a JNDI provider alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JNDI Settings**.
4. Locate the alias you want to delete and click the **×** icon in the **Delete** column. Integration Server displays a dialog box that prompts you to verify your action. Click **OK** to verify that you want to delete the JNDI provider alias.

Creating a JNDI Provider Failover List

You can configure Integration Server to automatically connect to an alternate JNDI provider if the connection to the primary JNDI provider becomes unavailable.

Use the **Edit JNDI Provider Alias** page to create a JNDI provider failover list containing alternate URLs that Integration Server can use if the primary connection fails.

Keep the following points in mind when adding JNDI provider URLs to the failover list:

- The JNDI providers must be the same type as the primary provider. For example, if the primary provider is a webMethods Broker, then the JNDI providers in the failover list must also be webMethods Brokers.
- The administered objects on the providers must be identical to the each other.
- Once Integration Server connects to a JNDI provider, it continues to use that JNDI provider until the connection is lost or interrupted.
- When you start or restart a connection alias, Integration Server attempts to connect to the primary JNDI provider. If the connection fails, Integration Server immediately attempts to connect to the first JNDI provider in the failover list. If the connection fails, Integration Server attempts to connect to the next JNDI provider in the list.
- When using webMethods Broker as a JNDI provider, you can keep objects in sync between webMethods Brokers using a webMethods Broker territory. That way objects can automatically forward from on webMethods Broker to another within the territory.
- When using a cluster of Universal Messaging realm servers as the JNDI provider, Software AG recommends that you do not specify a **Provider URL Failover List** value. The realm URLs specified in **Provider URL** function as the failover list.

Including SSL Configuration Information in the JNDI Provider Alias

When the JMS provider requires a one or two-way SSL connection with the JMS client, you must configure SSL certificate information on Integration Server. Many JMS providers have a set of properties that the JMS client uses to pass on the certificate details needed to establish an SSL connection with the JMS provider. Although these properties can be added to the `jndi.properties` file, you can include these details in the JNDI provider alias instead. This approach is only applicable when the JNDI provider and JMS provider use the same set of certificates such as when using Universal Messaging as the JMS and JNDI providers.

Including the SSL configuration in the JNDI provider alias, does the following:

- Secures the connection between Integration Server and the JNDI provider.
- Adds certificate information to the JNDI context. Integration Server passes the SSL certificate information into the JNDI context when the context is created.

When Universal Messaging is the JMS provider and the JNDI provider is on the same realm or cluster as Universal Messaging, the JMS connection factory can access the certificate information when creating the connection to the JMS provider. Other JMS providers may use the certificate information in the JNDI context in a similar fashion. Refer to the JMS provider documentation for more information.

SSL certificate information can be set using the `javax.net.ssl` properties for the Integration Server JVM and can be set securely using the **JVM Keystore Alias** and **JVM Truststore Alias** fields on the **Security > Certificates** page in Integration Server Administrator. However, providing truststore and keystore information in the JNDI provider alias is also a secure way of providing certificate information. Additionally, including certificate information in the alias allows the use of different certificates for establishing connections to JMS providers. Some organizations maintain multiple keystores and truststores for different applications or different departments.

Note:

Some JMS providers require that the SSL settings of the JVM for the JMS client must be used to establish the SSL connection. For these JMS providers, any information provided in the JNDI provider alias will be ignored. Refer to your JMS provider documentation for more information.

Adding SSL certificate information to the JNDI provider alias consists of the following basic steps:

1. Determine if the JMS provider supports including SSL certificate information in the JNDI provider properties or if the JMS provider requires that the truststore and keystore information be set using the `javax.net.ssl` properties in the JMS client JVM. If the JMS provider requires that the JVM properties be used, adding truststore and keystore information to the JNDI provider alias will have no effect.
2. Determine if one-way SSL or two-way SSL is required.
3. Make sure that a truststore alias exists for the truststore that contains the Certificate Authority (CA) certificates for the JNDI and JMS provider.

This step is required for one-way SSL and two-way SSL. For more information about truststores and truststore aliases, see [“Using Keystores and Truststores with Integration Server” on page 477](#).

4. If two-way SSL is required, make sure that a keystore alias exists for the keystore that contains the client certificates to use to connect to the JNDI and JMS providers.
5. Exchange key and certificate information with the JNDI and JMS providers.

Integration Server must have copies of the JNDI and JMS providers' public key and signing CA certificates. This is required for one-way or two-way SSL. For two-way SSL, you must give the JNDI and JMS providers the public key and CA certificate that will be used to establish a secure connection with the JNDI provider and JMS provider.

6. Identify the JMS provider-specific properties that must be populated in the JNDI context. For example, the properties for Universal Messaging are:

```
nirvana.ssl.keystore.path
nirvana.ssl.keystore.pass
nirvana.ssl.keystore.cert
nirvana.ssl.truststore.path
nirvana.ssl.truststore.pass
nirvana.ssl.protocol
```

7. In the JNDI provider alias, select **Use SSL**.
8. Use the following table to provide truststore and key alias information along with the JNDI property names obtained from the JMS provider.

Under Settings

In **Truststore Alias**, select the truststore alias you created in step 3. This is required for one-way or two-way SSL.

In **Keystore Alias**, select the keystore alias you created in step 4. This is required for two-way SSL.

In **Key Alias**, select the key alias you created in step 4.

Under JNDI Property Names

Provide values for the following properties:

- **Truststore Property Name**
- **Truststore Password Property Name**

Provide value for the following properties:

- **Keystore Property Name**
- **Keystore Password Property Name**

Optionally, provide values for **Keystore Format Property Name**.

Optionally, provide a value for **Private Key Property Name**.

This property is optional when using Universal Messaging as the JMS provider.

Performing a Test Lookup for a JNDI Provider

Using Integration Server Administrator, you can perform a test lookup for the primary JNDI provider and any failover JNDI providers specified in the **Provider URL Failover List**. Test lookup verifies that the URLs are valid.

Note:

Test lookup only verifies the validity of the URLs; it does not check or compare Objects in the JNDI namespace referenced by the specified provider URL aliases.

➤ To perform a test lookup for a JNDI provider

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JNDI Settings**.
4. Locate the alias you want to test and click ► in the **Test Lookup** column.

Integration Server returns a list of the Objects located in the JNDI namespace.

Note:

The list displays the Objects in the JNDI namespace referenced by the first successful provider alias URL lookup. Therefore, the results displayed may or may not be from the primary alias, they could be from a failover alias.

JNDI Provider Cache and Timeout Behavior for Administered Objects

Integration Server retrieves administered objects through JNDI when a JMS connection alias is enabled. Integration Server does this by instantiating a new context object using the specified JNDI provider. Integration Server employs this context to look up the connection factory and destinations specified by the JMS triggers, message sending services, and message receiving services that use the JMS connection alias.

Most JNDI provider implementations, including the webMethods Broker implementation, automatically cache administered objects locally when the context is instantiated. There are some JNDI providers, however, in which the caching and timeout behavior for administered objects is configurable on the provider. In most cases, using this type of provider is not an issue because the destination is cached by Integration Server when it is looked up. Integration Server only needs to look it up one time, and then all subsequent lookups are retrieved locally. However, if the administered objects are not cached in the context or if they have timed out, then a lookup request may require a new connection to the JNDI provider.

In this case, if the JNDI provider becomes unavailable after enabling the connection alias but before retrieving the destination, a `NamingException` can occur. The JNDI failover functionality in Integration Server addresses this situation. If Integration Server is unable to retrieve the destination from its original context, it will create a new context from one of the failover JNDI providers.

Keep in mind that, because Integration Server caches the destinations locally, it is possible that the cached destinations may become stale. In this case, the original destination will be used. You can reload administered objects by enabling and disabling the connection alias or monitoring the connection object for changes. If the administered objects are no longer usable, a `ServiceException` will occur. For more information about monitoring connections, see [“Monitoring a Connection Factory Object for Changes” on page 309](#).

Working with JMS Connection Aliases

A JMS connection alias specifies the information that Integration Server needs to establish an active connection between Integration Server and the JMS provider. Integration Server uses a JMS connection alias to send messages to and receive messages from the JMS provider.

Connecting to webMethods Broker with the Native webMethods API

When you use webMethods Broker as the JMS provider, you can configure a JMS connection alias to use the native webMethods API. The native webMethods API provides a way to connect the Integration Server directly to the webMethods Broker acting as the JMS provider. This eliminates the need for connection factories. Destinations can be created directly on the Broker and do not need to be stored in a JNDI provider. Consequently, you do not need to use a JNDI provider if all JMS connection aliases specify that the connection to webMethods Broker should be made using the native webMethods API.

If you choose to use webMethods Broker but do not want to connect natively (using the native webMethods API), you still need to use a JNDI provider.

Note: webMethods Broker is deprecated. Consequently, the ability to configure JMS connection alias to use the native webMethods API is deprecated as well.

Predefined JMS Connection Aliases

Integration Server includes predefined JMS connection aliases. Integration Server creates the aliases the first time Integration Server starts. If the aliases existed in an Integration Server that was migrated from a previous version, Integration Server keeps the existing aliases. That is, Integration Server does not overwrite the previous alias definition.

Integration Server has the following predefined JMS connection aliases:

JMS Connection Alias Description

DEFAULT_IS_JMS_CONNECTION	A JMS connection alias that defines a connection to Universal Messaging using the predefined JNDI provider alias DEFAULT_IS_JNDI_PROVIDER.
---------------------------	--

Integration Server configures this JMS connection alias when you launch the Integration Server the first time.

This connection alias is disabled by default.

PE_NON_TRANSACTIONAL_ALIAS	A JMS connection alias for the Process Engine that establishes a connection to Universal Messaging using the predefined JNDI provider alias DEFAULT_IS_JNDI_PROVIDER.
----------------------------	---

Integration Server configures this alias when you launch your Integration Server the first time.

Note: Integration Server creates this alias even if the Process Engine is not installed.

Note:

If you migrated to Integration Server 9.10 or later from an earlier version, you might have the EventBus alias which was a pre-defined JMS connection alias in versions prior to 9.10. This EventBus alias is for the EDA Event Bus server and defines a connection to Universal Messaging using the predefined JNDI provider alias DEFAULT_IS_JNDI_PROVIDER.

Creating a JMS Connection Alias

When you create a JMS connection alias, keep the following points in mind:

- You can use JNDI to retrieve administered objects (Connection Factories and Destinations) and then use the Connection Factory to create a connection. If you intend to use a JNDI provider,

you need to configure one or more JNDI provider aliases before creating a JMS connection alias.

- or -

You can use the native webMethods API to create the connection directly on the webMethods Broker. Because you can create Destinations at the Broker, you do not need to configure a JNDI provider alias if you intend to use the native webMethods API.

Note that if you elect to use the webMethods Broker but do not want to connect natively, you will need to use a JNDI provider and configure one or more JNDI provider aliases.

- Each JMS connection alias has an associated transaction type. Within Integration Server, certain functionality must be completed within a non-transacted session. For example, to use Integration Server to send or receive large message streams, you must specify a JMS connection alias whose transaction type is set to NO_TRANSACTION.
- When configuring a JMS connection alias from an on-premise Integration Server to a Universal Messaging server in the cloud, you need to modify the `custom_wrapper.conf` configuration file to include `com.softwareag.um.jndi.cf.url.override=true`. Specifically, modify the `custom_wrapper.conf` file to include:

```
wrapper.java.additional.n=-Dcom.softwareag.um.jndi.cf.url.override=true
```

Where *n* is the next available `wrapper.java.additional` number.

➤ To create a JMS connection alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. Click **Create JMS Connection Alias**.
5. Set the following **General Settings** for the JMS connection alias:

For this field	Specify
Connection Alias Name	Name of the connection alias. Each connection alias represents a connection factory to a specific JMS provider.
Description	A description of the JMS connection alias.
Transaction Type	Whether sessions that use this JMS connection alias will be transacted.
	<div style="display: flex; justify-content: space-between;"> Select To </div>

For this field	Specify
	<p data-bbox="600 252 1487 325">NO_TRANSACTION Indicate that sessions that use this JMS connection alias are not transacted.</p> <p data-bbox="600 346 1487 451">LOCAL_TRANSACTION Indicate that sessions that use this JMS connection alias are part of a local transaction.</p> <p data-bbox="600 472 1487 598">XA_TRANSACTION Indicate that sessions that use this JMS connection alias are part of an XA transaction.</p>
Connection Client ID	<p data-bbox="600 609 1487 682">The JMS client identifier associated with the connections established by this JMS connection alias.</p> <div data-bbox="600 693 1487 966" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="600 693 1487 735">Note:</p> <p data-bbox="600 735 1487 966">The connection client ID can be specified in the JMS connection alias or in the Connection Factory. If the connection client ID is specified in both places, Integration Server uses the value from the Connection Factory. For more information about the connection client ID, see “About the Connection Client ID” on page 298.</p> </div>
User (optional)	<p data-bbox="600 976 1487 1050">User name needed to acquire a connection from the connection factory.</p> <p data-bbox="600 1071 1487 1197">For more information about whether or not the JMS provider requires a user name and password to obtain a connection, refer to the product documentation for the JMS provider.</p>
Password (optional)	<p data-bbox="600 1207 1487 1281">Password needed to acquire a connection from the connection factory.</p> <p data-bbox="600 1302 1487 1400">For more information about whether or not the JMS provider requires a user name and password to obtain a connection, refer to the product documentation for the JMS provider.</p>

6. In the **Create Connection Using** list, select one of the following to indicate how administered objects (connection factories and destinations) will be looked up:
 - If you intend to use a JNDI provider, select **JNDI Lookup**.
 - If you intend to use the native webMethods API to create the connection directly on the webMethods Broker, select **Native webMethods API**.
7. If you selected **JNDI Lookup** in the **Create Connection Using** list, do the following in the remaining fields under **Connection Protocol Settings**:

For this field	Specify
JNDI Provider Alias Name	<p>The alias to the JNDI provider that you want this JMS connection alias to use to look up administered objects. For information about creating a JNDI provider alias, see “Creating a JNDI Provider Alias” on page 275.</p>
Connection Factory Lookup Name	<p>The lookup name for the connection factory that you want to use to create a connection to the JMS provider specified in this JMS connection alias.</p> <p>If the JMS connection alias connects to Universal Messaging, specify the Universal Messaging connection factory that you created when you set up your environment.</p> <p>If you are using SonicMQ as the JMS provider, specify the lookup name that refers to the serializable Java object file containing the SonicMQ object definitions. Include the .sjo extension as part of the lookup name.</p>
Create Administered Objects On Demand (Universal Messaging)	<p>Whether Integration Server creates administered objects on the JNDI provider if the object is not found when Integration Server looks up the object.</p> <p>Select the check box if you want Integration Server to create a destination or connection factory when an JNDI lookup for the object fails. For more information about creating administered objects on demand, see “Creating Administered Objects” on page 307.</p> <p>To create the administered objects on demand, the JNDI provider must be the Universal Messaging JNDI provider and the JMS provider must be Universal Messaging.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note:Software AG recommends against using this feature in a production environment. That is, leave the Create Administered Objects On Demand check box cleared for an Integration Server running in production.</p> </div>
Enable Follow the Master (Universal Messaging)	<p>Whether connections created from this JMS connection alias always connect to the master realm server in the Universal Messaging cluster. This setting affects producer and consumer connections created using this JMS connection alias. Do one of the following:</p> <ul style="list-style-type: none"> ■ Select the Enable Follow the Master check box to indicate that Integration Server always connects to the master realm server in a Universal Messaging cluster when Integration Server uses this JMS connection alias to send or receive messages.

For this field	Specify										
	<ul style="list-style-type: none"> ■ Clear the Enable Follow the Master check box to disable the follow the master behavior for this JMS connection alias when Integration Server uses this JMS connection alias to send or receive messages. <p>If your solution uses the round-robin feature for sending JMS messages, disable follow the master for the JMS connection alias.</p> <p>Note: Enable Follow the Master option is available to JMS connection aliases that use Universal Messaging as the JMS provider.</p>										
<p>Monitor webMethods Connection Factory</p>	<p>How Integration Server monitors the connection factory object for changes, if at all. This only applies if a JMS connection alias connects to the webMethods Broker using a webMethods Connection Factory object in a JNDI server.</p> <table border="1" data-bbox="617 840 1487 1522"> <thead> <tr> <th data-bbox="617 840 893 892">Select</th> <th data-bbox="893 840 1487 892">To</th> </tr> </thead> <tbody> <tr> <td data-bbox="617 892 893 1029">No</td> <td data-bbox="893 892 1487 1029">Indicate that Integration Server will not monitor the connection factory. This is the default.</td> </tr> <tr> <td data-bbox="617 1029 893 1134">Poll for changes (specify interval)</td> <td data-bbox="893 1029 1487 1134">Monitor the connection factory by polling for changes at an interval that you specify.</td> </tr> <tr> <td data-bbox="617 1134 893 1428">Poll for changes (interval defined by webMethods Connection Factory)</td> <td data-bbox="893 1134 1487 1428">Monitor the connection factory at an interval determined by the refresh interval specified for the webMethods Connection Factory object. For more information about configuring a cluster connection factory, see <i>Administering webMethods Broker</i> and <i>webMethods Broker Messaging Programmer's Guide</i>.</td> </tr> <tr> <td data-bbox="617 1428 893 1522">Register change listener</td> <td data-bbox="893 1428 1487 1522">Monitor the connection factory by registering an event listener.</td> </tr> </tbody> </table>	Select	To	No	Indicate that Integration Server will not monitor the connection factory. This is the default.	Poll for changes (specify interval)	Monitor the connection factory by polling for changes at an interval that you specify.	Poll for changes (interval defined by webMethods Connection Factory)	Monitor the connection factory at an interval determined by the refresh interval specified for the webMethods Connection Factory object. For more information about configuring a cluster connection factory, see <i>Administering webMethods Broker</i> and <i>webMethods Broker Messaging Programmer's Guide</i> .	Register change listener	Monitor the connection factory by registering an event listener.
Select	To										
No	Indicate that Integration Server will not monitor the connection factory. This is the default.										
Poll for changes (specify interval)	Monitor the connection factory by polling for changes at an interval that you specify.										
Poll for changes (interval defined by webMethods Connection Factory)	Monitor the connection factory at an interval determined by the refresh interval specified for the webMethods Connection Factory object. For more information about configuring a cluster connection factory, see <i>Administering webMethods Broker</i> and <i>webMethods Broker Messaging Programmer's Guide</i> .										
Register change listener	Monitor the connection factory by registering an event listener.										
<p>Polling Interval (minutes)</p>	<p>The number of minutes between polling attempts. The polling interval must be a positive integer. The default value is 60 minutes.</p> <p>Note: This field is only available if you selected Poll for changes (specify interval).</p>										
<p>Note:</p>	<p>For more information, see “Monitoring a Connection Factory Object for Changes” on page 309.</p>										

8. If you selected **Native webMethods API** in the **Create Connection Using** list, do the following to configure the connection to the Broker Server that you want to use as the JMS provider for this JMS connection alias:

For this field	Specify
Broker Host	Name (<i>DNSname:port</i> or <i>ipaddress:port</i>) of the machine on which the Broker Server resides.
Broker Name	Name of the webMethods Broker as defined on the Broker Server. The default name is Broker #1 .
Client Group	The name of the client group to which you want Integration Server to belong when it acts as a JMS client. The client group that you specify must already exist on the Broker Server. The default is IS-JMS.
Broker List	<p>A comma delimited list of Broker Servers on which the connection between the Integration Server (acting as the JMS client) and the Broker (acting as the JMS provider) can exist. This provides connection failover. If a connection failure occurs to the first Broker Server in the list, a connection attempt will be made to the next Broker Server listed. Use the following format for each webMethods Broker:</p> <pre>{webMethods Broker Name}@<host>[:port]</pre> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If a connection to a Broker Server is already configured (via Settings > Messaging > webMethods Messaging Settings), Integration Server populates the fields under Connection Protocol Settings with information about that Broker Server. If that Broker Server functions as the JMS provider, you may not need to edit any information. However, make sure that the Client Group field specifies the client group to which you want Integration Server to belong when it functions as a JMS client.</p> </div>
Keystore	<p>The full path to this Integration Server's keystore file. A keystore file contains the credentials (private key/signed certificate) that an entity needs for SSL authentication. If the Broker Server requires an SSL connection, then the information in this file is used to authenticate the Integration Server client to that Broker Server.</p> <p>The Integration Server's keystore file is stored on the machine on which the Integration Server resides.</p>
Keystore Type	The file type of the Integration Server's keystore file, which can be either PKCS12 or JKS .
Truststore	The full path to this Integration Server client's truststore file. A truststore file contains trusted root certificates for the authorities responsible for signing SSL certificates. For an SSL connection to be

For this field	Specify
	made, a valid trusted root for the SSL certificate stored in the keystore must be accessible in a truststore file.
	The Integration Server's truststore file is stored on the machine on which the Integration Server resides.
Truststore Type	The file type of the Integration Server's truststore file, which is JKS .
Encryption	Specify whether or not to encrypt the connection between the Integration Server and the webMethods Broker.

9. Under **Advanced Settings**, specify the following information for the JMS connection alias:

For this field	Specify
Class Loader	<p>The name of the class loader that you want to use with this JMS connection alias. Integration Server will use the specified class loader when performing certain activities with the JMS connection alias (send a message, receive a message, create a connection, create a destination, etc.)</p> <p>By default, Integration Server uses the server class loader. However, you can specify the class loader for a package instead. This may be helpful when working with third party JMS providers. For example, you might place the third party jars needed for each JMS provider in separate packages, specifically, in the <i>Integration Server_directory/instances/instance_name/packages/packageName/code/jars</i> directory. This can help prevent conflicts between the jars required for different JMS providers.</p>
Maximum CSQ Size (messages)	<p>The maximum number of messages that can exist in the client side queue for this JMS connection alias. Integration Server writes messages to the client side queue if the JMS provider is not available when messages are sent. Each JMS connection alias has its own client side queue.</p> <p>Specify -1 if you want the client side queue to be able to contain an unlimited number of messages. That is, specify -1 if you do not want to set a maximum limit.</p> <p>If you specify 0, Integration Server will not write messages to the client side queue for this JMS connection alias.</p>
Drain CSQ in Order	Whether Integration Server drains the client side queue by sending the messages to the JMS provider in the same order in which Integration Server placed the messages in the client side queue.

For this field	Specify
	<p>Select the check box if you want Integration Server to send messages from the client side queue in the same order in which Integration Server originally placed the messages in the client side queue.</p> <p>When the Drain CSQ in Order check box is selected, after the connection to the JMS provider is re-established, Integration Server continues to write new messages to the client side queue until the client side queue is completely drained. If the Drain CSQ in Order check box is not selected, after the connection to the JMS provider is re-established, Integration Server sends new messages directly to the JMS provider while it drains the client side queue.</p> <p>Note: You can also specify the number of messages Integration Server retrieves from the client side queue for delivery to the JMS provider at one time. By default, Integration Server sends 25 messages at a time. For more information about the <code>watt.server.jms.csq.batchProcessingSize</code> property, see “Server Configuration Parameters” on page 1017.</p>
<p>Create Temporary Queue</p>	<p>Whether Integration Server creates a temporary queue on the JMS provider to handle request-reply operations that do not specify a <code>replyTo</code> destination.</p> <p>Select the check box if you want Integration Server to create a temporary queue. Clear the check box if you do not want Integration Server to create a temporary queue.</p> <p>You must select the Create Temporary Queue check box if you want to select the Enable Request-Reply Listener for Temporary Queue check box.</p>
<p>Enable Request-Reply Listener for Temporary Queue</p>	<p>Specifies whether or not Integration Server creates a single dedicated <code>MessageConsumer</code> for receiving synchronous replies delivered to the temporary queue for this JMS connection alias. When this check box is cleared, Integration Server creates a new JMS <code>MessageConsumer</code> for each reply message. In many situations, creating one <code>MessageConsumer</code> per response does not impact performance. However, in some situations, such as when many threads invoke <code>pub.jms:sendAndWait</code> concurrently, creating a <code>MessageConsumer</code> for every expected response can impact performance. When this check box is selected, Integration Server creates a dedicated consumer for receiving replies to requests published using this JMS connection alias. For more information about creating a dedicated listener for receiving replies, see “Creating a Dedicated Listener for Receiving Replies” on page 299.</p>

For this field	Specify
Enable Destination Management with Designer (Broker and Universal Messaging)	Whether users can use Designer to create, list, and modify destinations on the webMethods Broker or when working with JMS triggers.
	Select the check box if you want Designer users to be able to create, list, and modify destinations using a JMS trigger that uses this JMS connection alias.
	Software AG recommends that you prevent Designer users from managing destinations in a production environment. For more information about using Designer to manage destinations, see “Allowing Destinations to Be Managed through the JMS Connection Alias and Designer” on page 296.
Create New Connection per Trigger	Whether Integration Server creates a separate connection to the JMS provider for each JMS trigger.
	Select the check box if you want Integration Server to create a separate connection for each JMS trigger that uses this JMS connection alias.
	If you want a concurrent JMS trigger that uses this JMS connection alias to use multiple connections to the JMS provider, you must configure the alias to create a separate connection per trigger. For more information, see “Allowing Multiple Connections for a JMS Connection Alias” on page 297.

10. Under **Producer Caching**, specify the following to configure pools for caching of JMS Session and MessageProducer objects. For more information about producer caching, see [“Configuring Producer Caching for Sending JMS Messages”](#) on page 300.

For this field	Specify						
Caching Mode	Whether to enable caching of JMS Session and MessageProducer objects for this connection alias.						
	<table border="1"> <thead> <tr> <th>Select...</th> <th>To...</th> </tr> </thead> <tbody> <tr> <td>DISABLED</td> <td>Indicate that Integration Server does not cache JMS Session or MessageProducer objects.</td> </tr> <tr> <td>ENABLED PER DESTINATION</td> <td>Enable caching of JMS Session and MessageProducer objects. For a non-transacted JMS connection alias, Integration Server caches a</td> </tr> </tbody> </table>	Select...	To...	DISABLED	Indicate that Integration Server does not cache JMS Session or MessageProducer objects.	ENABLED PER DESTINATION	Enable caching of JMS Session and MessageProducer objects. For a non-transacted JMS connection alias, Integration Server caches a
	Select...	To...					
DISABLED	Indicate that Integration Server does not cache JMS Session or MessageProducer objects.						
ENABLED PER DESTINATION	Enable caching of JMS Session and MessageProducer objects. For a non-transacted JMS connection alias, Integration Server caches a						

For this field	Specify
	<p>Session object and a MessageProducer object.</p> <p>For a transacted JMS connection alias, Integration Server caches a Session object.</p>
	<p>Note: Integration Server supports session caching for transacted JMS connection aliases when the JMS provider is Universal Messaging or WebSphere MQ 7.5. For information about Universal Messaging versions supported with this version of Integration Server, see “Supported JMS Providers” on page 314.</p>
<p>Minimum Session Pool Size (unspecified destinations)</p>	<p>The minimum number of entries in the default session pool. The default is 1.</p>
<p>Maximum Session Pool Size (unspecified destinations)</p>	<p>The maximum number of entries in the default session pool. The default is 30.</p>
<p>Minimum Pool Size per Destination</p>	<p>The minimum number of entries in each destination-specific pool.</p>
<p>Maximum Pool Size per Destination</p>	<p>The maximum number of entries in each destination-specific pool. A value of 0 (or blank) indicates that Integration Server does not create separate pools for any of the destinations associated with the JMS connection alias.</p>
<p>Destination Lookup Name List</p>	<p>A semicolon delimited list of the lookup names for the destinations for which you want Integration Server to create separate pools.</p>
<p>Queue List</p>	<p>Note: This field appears only when the JMS connection alias specifies JNDI Lookup for creating the connection to the JMS provider.</p>
<p>Queue List</p>	<p>A semicolon delimited list of the queues for which you want Integration Server to create separate pools.</p>
<p>Queue List</p>	<p>Note: This field only appears when the JMS connection alias specifies Native webMethods API for creating the connection to the webMethods Broker.</p>
<p>Topic List</p>	<p>A semicolon delimited list of the topics for which you want Integration Server to create separate pools.</p>
<p>Topic List</p>	<p>Note:</p>

For this field	Specify
	This field only appears when the JMS connection alias specifies Native webMethods API for creating the connection to the webMethods Broker.
Idle Timeout	<p>The number of milliseconds before Integration Server removes an inactive pool entry. The timeout value applies to the default session pool and the destination-specific pools.</p> <p>A value of 0 indicates that pool entries never expire. A value of -1 indicates that Integration Server uses the system default as defined by the <code>watt.server.jms.producer.pooledSession.timeout</code> parameter.</p> <p>This default is 60000 milliseconds</p>

11. Under **Producer Retry**, specify the following to configure automatic retry of `pub.jms:send` services that use this JMS connection alias to send a message to the JMS provider. For more information about automatic retry, see [“Configuring Automatic Retry when Sending JMS Messages Using the `pub.jms:send` Service” on page 302](#).

For this field	Specify
Max Retry Count	The maximum number of times that Integration Server will automatically retry a <code>pub.jms:send</code> service that fails because of a transient error. A value of 0 indicates that automatic retry is disabled for this JMS connection alias. The default is 0.
Retry Interval (milliseconds)	The number of milliseconds that Integration Server waits between retry attempts. The default is 1000 milliseconds (1 second).

Note:

If the JMS connection alias is transacted or uses a connection factory to which the multi-send guaranteed policy is applied, Integration Server ignores the producer retry values.

12. Under **Enhanced Logging**, specify the following to configure additional logging for the sending and/or receiving of JMS messages that use this messaging connection alias.

For this field	Specify
Logging Type	<p>Where Integration Server writes log messages when enhanced logging is enabled for the message producers and/or consumers that use this JMS connection alias to send and/or receive messages.</p> <ul style="list-style-type: none"> ■ SERVER LOG. Write enhanced logging messages to the server log. If you specify the server log as the destination, make sure

For this field	Specify
	<p>to increase the logging level for the 0168 Messaging (Enhanced Logging) server log facility to at least Info.</p> <ul style="list-style-type: none"> ■ MESSAGING AUDIT LOG. Write enhanced logging messages to the messaging audit log. <p>You can select one of the options only. Integration Server cannot write enhanced logging messages to the server log <i>and</i> the messaging audit log.</p>
Enable Producer Message ID Tracking	Select to indicate that Integration Server writes additional log messages when a message producer uses this JMS connection alias to send messages to a destination in Producer Message ID Tracking: Include Destinations .
Producer Message ID Tracking: Include Destinations	The destination names for which Integration Server performs additional logging when sending messages to the destination. Use a semicolon (;) to separate each destination name. Leave this field blank if you want Integration Server to perform enhanced logging for every destination to which this JMS connection alias sends messages.
Enable Consumer Message ID Tracking	Select to indicate that Integration Server writes additional log messages for messaging consumers (triggers) that use this JMS connection alias to receive messages. Integration Server writes additional log message for the JMS triggers listed in Consumer Message ID Tracking: Include Triggers .
Consumer Message ID Tracking: Include Triggers	The fully qualified name of the JMS triggers for which Integration Server performs additional logging during trigger processing. Use a semicolon (;) to separate each trigger name. Leave this field blank if you want Integration Server to perform enhanced logging for every JMS trigger that uses this JMS connection alias to receive messages.

For more information about using enhanced logging for sending and receiving of JMS messages, [“Using Enhanced Logging for Messaging” on page 335](#).

13. Click **Save Changes**.

14. Enable the JMS connection alias.

Allowing Destinations to Be Managed through the JMS Connection Alias and Designer

You can configure the JMS connection alias to enable Designer users to manage destinations and durable subscribers when working with JMS triggers. When the JMS connection alias is configured for managing destinations (i.e., the **Enable Destination Management with Designer** check box

is selected), you can use Designer to create and modify destinations and durable subscribers on webMethods Broker or Software AG Universal Messaging.

Note:

The ability to use Designer to manage destinations on Broker or Universal Messaging is a design-time feature. In a production environment, this functionality should be disabled.

To manage destinations on Broker, the following must be true

- The JMS connection alias must use Broker as the JMS provider.
- Broker must be Broker version 7.1 or higher.
- The versions of following three Broker jar files installed on Integration Server must be the 8.0 SP1 or higher versions of the files.
 - *Software AG_directory /common/lib/wm-brokerclient.jar*
 - *Software AG_directory /common/lib/wm-jmsclient.jar*
 - *Software AG_directory /common/lib/wm-jmsnaming.jar*

To manage destinations on Universal Messaging, the following must be true:

- The JMS connection alias must use Universal Messaging as the JMS provider.
- The Universal Messaging version must be supported with this version of Integration Server. For information about supported versions, see [“Supported JMS Providers” on page 314](#).

For more information about using Designer to modify Destinations on the Broker or Universal Messaging, see *webMethods Service Development Help*.

Allowing Multiple Connections for a JMS Connection Alias

You can configure a JMS connection alias to instruct Integration Server to create a separate connection to the JMS provider for each JMS trigger that uses the alias. Creating separate connections for individual triggers can improve performance, especially when processing a high volume of messages across many triggers.

By default, an alias creates a single connection to the JMS provider, and each JMS trigger that uses the alias shares the same connection. Integration Server uses this same connection to send JMS messages when executing `pub.jms*` services that specify the JMS connection alias.

If you select the **Create New Connection per Trigger** check box when configuring a JMS connection alias, Integration Server creates a new connection to the JMS provider for each JMS trigger that uses the alias. These connections are in addition to the connection that Integration Server uses for sending JMS messages. Therefore, if a JMS connection alias is associated with 3 JMS triggers, there will be a total of 4 connections associated with the alias.

When an alias creates a separate connection for each trigger, you can configure the associated concurrent JMS triggers to obtain multiple connections to the JMS provider, thereby noticeably improving trigger throughput. Keep in mind, however, that each connection used by the trigger requires a dedicated Integration Server thread, regardless of the current throughput.

Note:

If you select the **Create New Connection per Trigger** check box, then the **Ignore Locally Published** feature on associated JMS triggers will not work. For the trigger to ignore locally published messages, the publisher and subscriber must share the same connection. When the JMS connection alias creates a new connection per trigger, the publisher and subscriber will not share the same connection.

Integration Server supports creating and using multiple connections for a single JMS connection alias with the supported JMS providers. For a list of supported JMS providers, see [“Supported JMS Providers” on page 314](#).

Note:

When Integration Server creates multiple connections, Integration Server uses the same client ID for each connection. While the webMethods Broker permits this, some JMS providers do not support the use of multiple connections with the same client ID or require additional configuration to support it. This can be particularly true when working with topics and/or durable subscribers. Review the JMS provider documentation before configuring the JMS connection alias or a JMS trigger to use multiple connections.

To use multiple connections for a single JMS connection alias when using webMethods Broker as the JMS provider, the following must be true:

- webMethods Broker must be webMethods Broker version 7.1 or higher.
- The versions of following three webMethods Broker jar files installed on Integration Server must be the 8.0 SP1 or higher versions of the files.
 - *Software AG_directory* /common/lib/wm-brokerclient.jar
 - *Software AG_directory* /common/lib/wm-jmsclient.jar
 - *Software AG_directory* /common/lib/wm-jmsnaming.jar

For more information about configuring JMS triggers, see the *webMethods Service Development Help* in Software AG Designer.

About the Connection Client ID

The connection client ID is the JMS client identifier associated with the connections established by a JMS connection alias. The connection client ID that Integration Server uses for a connection created from a JMS connection alias depends on one or more of the following:

- The value of the **Connection Client ID** field for the JMS connection alias.
- The connection client ID specified in the Connection Factory used by the JMS connection alias.
- Whether the JMS connection alias is configured to create a new connection for each JMS trigger. That is, whether or not the **Create New Connection per Trigger** check box is selected.

Using the above information, Integration Server determines the connection client ID as follows:

- If only the JMS connection alias specifies the connection client ID, Integration Server uses this value for any connections created from the alias. When connecting to the webMethods Broker

natively (the **Create Connection Using** list is set to **Native webMethods API**), Integration Server always uses the connection client ID from the JMS connection alias.

- If only the Connection Factory specifies the connection client ID, Integration Server uses this value for any connections created from the alias.
- If the JMS connection alias and the Connection Factory specify the connection client ID, Integration Server uses the value in the Connection Factory. This is true when working with all JMS providers, including the webMethods Broker.
- When the **Create New Connection per Trigger** check box is not selected, each JMS trigger that uses the JMS connection alias will use the same connection. Each connection will have the same connection client ID.
- When the **Create New Connection per Trigger** check box is selected, each JMS trigger that uses the JMS connection alias will create its own connection to the JMS provider.
 - If Integration Server uses the connection client ID in the Connection Factory, each connection for a JMS trigger will have the same connection client ID.
 - If Integration Server uses the connection client ID from the JMS connection alias, each connection for a JMS trigger will be unique to the JMS trigger. The connection client ID will consist of the value of the **Connection Client ID** field in the JMS connection alias plus the fully qualified name of the JMS trigger.

Note:

To receive messages in a load balanced manner, each JMS trigger must connect to the webMethods Broker using the same connection client ID. Because the **Create New Connection per Trigger** option can change the connection client ID, you must be sure that use of this option is consistent across all Integration Servers across the Integration Server group.

Creating a Dedicated Listener for Receiving Replies

Integration Server includes the ability to create a dedicated consumer for receiving replies to a published request. You configure this functionality per JMS messaging connection alias.

When the `pub.jms:sendAndWait` service executes a synchronous request-reply, Integration Server sends the request message to the JMS provider and waits for a reply. By default Integration Server creates a new JMS MessageConsumer for each reply message. In many situations, creating one MessageConsumer per response does not impact performance. However, in some situations, such as when many threads invoke `pub.jms:sendAndWait` concurrently, creating a MessageConsumer for every expected response can impact performance.

To address this, Integration Server includes an option named **Enable Request-Reply Listener for Temporary Queue** that, when selected, creates a single dedicated MessageConsumer for receiving synchronous replies sent by the JMS connection alias. If Integration Server uses a single consumer to retrieve all synchronous replies for requests sent by a particular JMS connection alias, when executing `pub.jms:sendAndWait` service, Integration Server assigns a unique value to the new `wm_tag` field used as a property in the JMS message, specifically `JMSMessage/properties/wm_tag`. When using `pub.jms:reply` to reply to a JMS message request for which `wm_tag` is populated, the

replying Integration Server maps the value of *wm_tag* to *JMSMessage/header/JMSCorrelationID* field in the reply message.

To use a dedicated consumer to retrieve replies to all requests sent using a particular JMS connection alias, the following must be true:

- The JMS connection alias used to send the request must have the following configured:
 - The **Create Temporary Queue** check box must be selected.
 - The **Enable Request-Reply Listener for Temporary Queue** check box must be selected. This is a new option for a JMS connection alias.
- The `pub.jms:sendAndWait` invocation:
 - Must be synchronous (the *async* input parameter must be set to false).
 - Must not specify a value for the *destinationNameReplyTo* input parameter.

Configuring Producer Caching for Sending JMS Messages

When sending a JMS message, Integration Server creates and closes a new JMS Session object and a JMS MessageProducer object for each message. This can introduce overhead for some JMS providers. To improve performance when sending JMS messages, you can configure producer-side pooling. For each JMS connection alias, Integration Server can create the following:

- A default session pool containing JMS Session objects. When a default session pool is defined for a JMS connection alias, Integration Server draws from a pool of open JMS Sessions for sending a JMS message instead of opening and closing a JMS Session for each JMS message. Integration Server uses the default session pool only when sending a message to a destination that does not have its own pool. When using the default session pool, Integration Server creates a new MessageProducer each time it sends a JMS message.
- Destination-specific pools for sending JMS messages to specific destinations. Integration Server creates a pool for each specified destination. The composition of a destination-specific pool varies depending on the transaction type of the JMS connection alias.
 - For a non-transacted JMS connection alias, an entry in a destination-specific pool consists of a Session object and a MessageProducer object. When sending a JMS message to one of the specified destinations, Integration Server uses a Session object and MessageProducer object from the pool instead of creating and closing a Session object and MessageProducer object for each JMS message.
 - For a transacted JMS connection alias, an entry in a destination-specific pool contains a Session object. When sending a JMS message to one of the specified destinations, Integration Server uses a Session object from the pool instead of creating and closing a Session object for each JMS message. Integration Server creates a new MessageProducer each time it sends a JMS message.

Note: Integration Server supports session caching for transacted JMS connection aliases when the JMS provider is Universal Messaging or WebSphere MQ 7.5. For information

about Universal Messaging versions supported with this version of Integration Server, see [“Supported JMS Providers” on page 314](#).

You can specify the minimum and maximum sizes for the default session pool and all destination pools. Additionally, you can identify the destinations for which Integration Server creates specific pools. If the JMS connection alias specifies the use of a connection factory object to connect to the JMS provider, you specify a single list of destinations. If the JMS connection alias specifies the Native webMethods API for connecting to the webMethods Broker, you must specify separate lists for the queues and topics for which you want Integration Server to create destination pools.

Consider the following examples that explain how Integration Server creates session and destination pools based on the information specified for transacted and non-transacted connection aliases:

Example of creating session and destination pools for a non-transacted JMS connection alias

Suppose that a non-transacted JMS connection alias named "myAlias" connects to the webMethods Broker using the Native webMethods API and the fields are set as described in the following table.:

Field	Value
Minimum Pool Size	1
Maximum Pool Size	10
Minimum Pool Size per Destination	1
Maximum Pool Size per Destination	5
Queue List	myQueue1; myQueue2
Topic List	myTopic
Idle Timeout	70000

Using the above information, Integration Server creates a default session pool with a minimum size of 1 and a maximum size of 10. This pool contains JMS Session objects only. Integration Server uses an entry from the pool when sending a message to destination that does not have its own pool.

Integration Server also creates three destination pools: one each for the queues myQueue1 and myQueue2, and one for the topic myTopic. Each of these pools has a maximum size of 5 pool entries. Messages sent to the destinations myQueue1, myQueue2, or myTopic will use an entry (a Session object and MessageProducer object) from the pool created for the destination. Messages sent to other destinations will use a Session from the default session pool.

An entry in the default or destination-specific pools expires after the entry has been inactive for over 70000 milliseconds (70 seconds).

Example of creating session and destination pools for a transacted JMS connection alias

Suppose that a transacted JMS connection alias named "myAlias1" connects to WebSphere MQ 7.5 using JNDI and the fields are set as described in the following table:

Field	Value
Minimum Pool Size	1
Maximum Pool Size	15
Minimum Pool Size per Destination	1
Maximum Pool Size per Destination	10
Destination Lookup Name List	myQueue1; myQueue2; myTopic1
Idle Timeout	80000

Using the above information, Integration Server creates a default session pool with a minimum size of 1 and a maximum size of 15. This pool contains JMS Session objects only. Integration Server uses an entry from the pool when sending a message to destination that does not have its own pool.

Integration Server also creates three destination pools: one each for the queues myQueue1 and myQueue2, and one for the topic myTopic1. Each of these pools has a maximum size of 10 pool entries. Messages sent to the destinations myQueue1, myQueue2, or myTopic1 will use an entry (a Session object) from the pool created for the destination. Messages sent to other destinations will use a Session from the default session pool.

An entry in the default or destination-specific pools expires after the entry has been inactive for over 80000 milliseconds (80 seconds).

Configuring Automatic Retry when Sending JMS Messages Using the pub.jms:send Service

You can configure a JMS connection alias so that Integration Server automatically retries a pub.jms:send service that uses the JMS connection alias if the service fails because of a transient error. To configure automatic retry for instances of the pub.jms:send service that use a particular JMS connection alias to send messages to the JMS provider, you specify the following for the alias:

- **Maximum number of retry attempts.** The **Max Retry Count** field determines the maximum number of times that Integration Server will retry a particular pub.jms:send service. A **Max Retry Count** of 0 indicates that automatic retry is disabled for the JMS connection alias.
- **Interval between retry attempts.** The **Retry Interval** field determines the number of milliseconds that Integration Server waits between retry attempts. The default interval is 1000 milliseconds (1 second).

The JMS connection alias must also meet the following criteria for a `pub.jms:send` service that uses the alias to be retried after a transient error:

- The JMS connection alias must be enabled.
- The JMS connection alias must have a transaction type of `NO_TRANSACTION`. Integration Server will not retry a `pub.jms:send` service that is executed as part of a transaction.
- If the JMS connection alias specifies the Broker as the JMS provider, the JMS connection alias must not use a cluster connection factory to which the multisend guaranteed policy is applied.

The following table describes the retry process that Integration Server uses when the the JMS connection alias is configured for retry and the `pub.jms:send` service fails because of a transient error.

Stage	Description
1	Execution of the <code>pub.jms:send</code> service fails because of a transient error.
2	Integration Server waits the length of the retry interval. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: If Integration Server begins to shut down while waiting to retry, Integration Server interrupts the waiting period and retries the service.</p> <p>Note: If the JMS connection alias becomes disabled while Integration Server waits to retry, Integration Server interrupts the retry process. Depending on whether the client side queue is enabled for the JMS connection alias, either writes the JMS message to the client side queue or throws the original exception that caused the <code>pub.jms:send</code> service to fail. For more information, see the description for stage 5, which follows.</p> </div>
3	Integration Server re-executes the <code>pub.jms:send</code> service.
4	If the <code>pub.jms:send</code> service fails again because of a transient error Integration Server repeats steps 2 and 3 until one of the following occurs: <ul style="list-style-type: none"> ■ The <code>pub.jms:send</code> service executes successfully. ■ Retry failure occurs because the maximum number of retry attempts have been made.
5	If retry failure occurs, Integration Server will do one of the following depending on whether or not the JMS connection alias uses a client side queue: <ul style="list-style-type: none"> ■ If the client side queue is in use, Integration Server writes the message created by the <code>pub.jms:send</code> service to the client side queue.

Stage	Description
	<ul style="list-style-type: none">■ If the client side queue is not in use, Integration Server throws the original exception thrown by the JMS provider.

Note:

A client side queue is in use if the JMS connection alias has a **Maximum CSQ Size** value greater than 0 (zero) and the *useCSQ* input parameter is set to true for the `pub.jms:send` service.

About Retrying the `pub.jms:send` Service when Software AG Universal Messaging Is the JMS Provider

When the JMS connection alias is configured to retry the `pub.jms:send` service upon transient error and Universal Messaging is the JMS provider, Integration Server may make some changes to the local instance of the Universal Messaging connection factory to prevent or at least delay an exception from being thrown by Integration Server. Integration Server makes changes so that if Integration Server loses its connection to the Universal Messaging cluster, Integration Server does not throw an exception right away. Instead, Universal Messaging suppresses the exception for a period of time. During this time, Universal Messaging attempts to restore the cluster quorum. Concurrently, Integration Server attempts to re-establish a connection to Universal Messaging. During this delay, Integration Server is not notified of the exception and the JMS connection alias will not be stopped. However, JMS triggers that use the JMS connection alias will not receive any messages. Additionally, any `pub.jms:send` services that were in the midst of using the JMS connection alias to send a message will be retried based on the retry interval and retry count set for the JMS connection alias. If Universal Messaging cannot restore a cluster quorum, Integration Server throws the exception. At this point, any JMS trigger that uses the JMS connection alias will be stopped and any services that send JMS messages using the JMS connection alias will throw exceptions immediately. Integration Server then attempts to reconnect to Universal Messaging.

Integration Server makes changes to the Universal Messaging connection factory if the connection factory has a `MaxReconAttempts` property set to -1. (A value of -1 is the default and suggests that the `MaxReconAttempts` value has not been changed on Universal Messaging.) Integration Server makes the following changes to the local instance of the Universal Messaging connection factory:

- Sets `ConxExceptionOnFailure` to true.
- Sets `MaxReconAttempts` to 35.
- Sets `ReconnectInterval` to 2000 milliseconds.

Integration Server makes changes to the local instance of the Universal Messaging connection factory only. The `ConnectionFactory` will not be changed on the JNDI provider, which means that other clients that use the `ConnectionFactory` will not be impacted.

To prevent Integration Server from making changes to the local instance of the Universal Messaging connection factory, use the Universal Messaging Enterprise Manager to set the `MaxReconAttempts` property to a value greater than -1.

Editing a JMS Connection Alias

After you create a JMS connection alias, you might need to modify properties of the alias you created or a default alias. For example, you might want to reduce the amount of memory that the client side queue can occupy by decreasing the maximum number of messages that can be placed in the client side queue. You can edit any properties of a JMS connection alias with the exception of the alias name.

Keep the following information in mind when editing a JMS connection alias:

- You must disable a JMS connection alias before you can edit it.
- Because Integration Server uses the connection client ID as the first part of the durable subscriber name, changing the connection client ID results in the creation of durable subscribers for triggers that use the alias. The durable subscribers that use the old connection client ID as part of the name are not automatically removed from Universal Messaging. Old durable subscribers might contain messages and continue to receive messages.

➤ To edit a JMS connection alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. In the JMS Connection Alias Definitions list, select the JMS connection alias that you want to edit. Integration Server Administrator displays details about the connection alias.
5. Click **Edit JMS Connection Alias**.
6. Edit the properties of the connection alias. For more information about the fields, see [“Creating a JNDI Provider Alias” on page 275](#). Note that the **Connection Alias Name** field cannot be modified.
7. Click **Save Changes**.

Enabling and Disabling a JMS Connection Alias

When a JMS connection alias is enabled, Integration Server can use the alias to obtain connections, send messages, and receive messages on behalf of services and JMS triggers. When a connection alias is disabled, Integration Server suspends all JMS triggers that use the alias. Additionally, any services that use a disabled JMS connection alias to send or receive messages will end in error.

➤ To enable or disable a JMS connection alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. In the JMS Connection Alias Definitions list, do one of the following in the Enabled column:
 - Click **No** if the alias is disabled and you want to enable it.
If Integration Server cannot enable the alias, it displays a message under the alias indicating why Integration Server cannot enable the alias.
 - Click **Yes** if the alias is enabled and you want to disable it.

Deleting a JMS Connection Alias

Before you delete a JMS connection alias, make sure of the following:

- The JMS connection alias is disabled.
- No services or JMS triggers rely on the JMS connection alias. If a JMS trigger uses a JMS connection alias, Integration Server will prevent the JMS connection alias from being deleted.

> To delete a JMS connection alias

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. In the JMS Connection Alias Definitions list, disable the alias if it is not already disabled.
5. Locate the alias you want to delete and click **X** in the **Delete** column. Integration Server displays a dialog box that prompts you to verify your action. Click **OK** to verify that you want to delete the JMS connection alias.

Specifying a Connection Monitoring Period

Integration Server periodically checks the state of an active connection to the JMS provider. You can configure how frequently Integration Server checks the connection status using the `watt.server.jms.connection.monitorPeriod` property. The default is 45 seconds.

If the connection between Integration Server and the JMS provider fails, Integration Server attempts to re-establish the connection automatically after a 20 second delay. You can configure how long

Integration Server waits between attempts to re-establish the connection by changing the value of the `watt.server.jms.connection.retryPeriod` property

For more information about server configuration parameters, see [“Server Configuration Parameters” on page 1017](#).

Specifying a Retry Interval for Failed Connections

When a connection between Integration Server and the JMS provider fails, Integration Server attempts to re-establish the connection automatically after a 20 second delay. You can configure how long Integration Server waits between attempts to re-establish the connection by changing the value of the `watt.server.jms.connection.retryPeriod` property. For more information about this property, see [“watt.server.” on page 1053](#).

Specifying a Keep-Alive Interval

Integration Server periodically pings the JMS provider to keep the connection between the Integration Server and the JMS provider active. The ping acts as a keep-alive request. By default, Integration Server pings the JMS provider every 300 seconds. You can configure how frequently Integration Server pings the JMS provider by changing the value of the `watt.server.jms.connection.pingPeriod` property. For more information about this property, see [“watt.server.” on page 1053](#).

Creating Administered Objects

The JMS provider that you select should provide you with the tools to create and configure administered objects in a JNDI namespace. For most JMS providers, Integration Server cannot be used to create and configure administered objects. For more information about working with administered objects, refer to the product documentation for your chosen JMS provider.

However, Integration Server can create administered objects in the JNDI namespace when Universal Messaging is the JMS provider and the JNDI provider. Integration Server creates the administered object automatically when the lookup for the object fails. For example:

- If a JMS connection alias attempts to connect to Universal Messaging using a connection factory that does not exist, Integration Server creates the connection factory and adds it to the JNDI namespace on Universal Messaging. Integration Server copies the connection URL from the JNDI settings.
- If a `pub.jms*` service specifies a topic or queue as the message destination and the destination does not exist, Integration Server creates the topic or queue on Universal Messaging and then adds it to the JNDI namespace.
- If a JMS trigger subscribes to a topic or queue that does not exist, Integration Server creates the topic or queue on Universal Messaging and then adds it to the JNDI namespace.

Whether or not Integration Server creates a missing administered object automatically is configured per JMS connection alias using the **Create Administered Objects On Demand** check box.

If you configure a JMS connection alias to create destinations and connection factories on demand, you do not need to use each time you want to create a connection factory or destination. Nor do you need to use Designer to create a destination from which a JMS trigger can retrieve messages. Using Integration Server to create administered objects on demand provides convenience and possible time savings during the development cycle.

For example, suppose that you receive an Integration Server package that contains JMS triggers and services that publish JMS messages. Further suppose that your Integration Server contains JMS connection aliases with the same names as the JMS connection aliases used by the services and triggers in the package, the JMS connection aliases are configured to create administered objects on demand, and your Integration Server has a JNDI alias points to a running Universal Messaging server. When you start the JMS connection aliases, Integration Server creates any missing connection factories. As you use the sending services and JMS triggers in the package, Integration Server creates any missing destinations needed by the services and triggers.

When using Integration Server to create administered objects on demand, keep the following points in mind:

- To create the administered objects on demand, the JNDI provider must be the Universal Messaging JNDI provider and the JMS provider must be Universal Messaging. This because Integration Server uses the URL in the JNDI provider alias to create the objects in the JNDI namespace and on Universal Messaging.
- Many JNDI providers, including the Universal Messaging JNDI provider, handle connection factory names and destination names in a case sensitive way. That is, “myFactory” and MYFactory” would be two different objects.
- Integration Server provides the ability to create the objects only. Integration Server cannot update or delete the objects that it adds.
- When Integration Server creates a connection factory specified in a JMS connection alias with a transaction type of XA_TRANSACTION, Integration Server creates an XAConnectionFactory (javax.jms.XAConnectionFactory).

Note:

For a JMS connection alias with a transaction type of LOCAL_TRANSACTION, Integration Server creates a regular connection factory (javax.jms.ConnectionFactory).

- If a connection factory exists but does not match the transaction type used by the JMS connection alias, Integration Server does not create a new connection factory. For example, suppose the JMS connection alias specifies a transaction type of XA_Transaction and a Connection Factory Lookup Name of “myConnectionFactory”. Additionally, suppose that a javax.jms.ConnectionFactory object named “myConnectionFactory” already exists in the JNDI namespace. When Integration Server starts the JMS connection alias and looks up the connection factory in the JNDI namespace, Integration Server finds the existing “myConnectionFactory” javax.jms.ConnectionFactory object and does not delete and recreate the object to be a javax.jms.XAConnectionFactory.
- Integration Server will not automatically create a connection factory that uses the Horizontal Scalability feature offered by Universal Messaging. If you want a JMS connection alias to use a connection factory that leverages the functionality available in the Horizontal Scalability feature, you must create the connection factory in Universal Messaging.

- Integration Server simply creates the administered object if the object does not exist when Integration Server looks it up. If the name of the administered object is not specified correctly, adding objects on demand could result in the creation of unwanted administered objects. For example, if the **Create Administered Objects On Demand** check box is selected for a JMS connection alias, and that alias is used in `pub.jms:send` service that specifies “myQueue” as the destination, an instance of the `pub.jms:createConsumer` service that specifies “my_queue”, and a JMS trigger that specifies “MyQueue” as the destination, Integration Server creates three unique destinations in the JNDI namespace.

Note:Software AG recommends against using this feature in a production environment. That is, leave the **Create Administered Objects On Demand (Universal Messaging)** check box cleared for a JMS connection alias on an Integration Server running in production.

Monitoring a Connection Factory Object for Changes

When you create or edit the JMS connection aliases that use a connection factory object to establish a connection to the webMethods Broker, you can indicate whether or not Integration Server monitors the connection factory for changes.

When Integration Server uses a connection factory object to establish a connection to the webMethods Broker, Integration Server uses JNDI Lookup to create the connection. After establishing the connection, Integration Server normally does not lookup the connection factory object again until the connection is restarted. Consequently, in between connection restarts, the connection does not reflect any changes that may have been made to the connection factory object, such as a change to the cluster policy or a change to the webMethods Brokers in a cluster. By monitoring the connection factory, Integration Server can update the related connection automatically after the connection factory changes.

You can configure Integration Server to monitor the connection factory object in one of the following ways:

- **Poll for changes.** Integration Server periodically looks up the specified connection factory object from the JNDI provider and compares it to the connection factory object used to establish the connection. If Integration Server detects a change, it refreshes the connection automatically.
- **Listen for changes.** Integration Server registers an event listener with the JNDI provider to receive notifications about changes to the cluster connection factory. When Integration Server receives notification of a change it refreshes the connection automatically.

Regardless of the method used to monitor changes, Integration Server attempts to update and refresh the connection without interrupting any services sending a JMS message or triggers receiving a JMS message. The following sections provide more information about the monitoring options.

Note:webMethods Broker is deprecated.

Polling for Changes

You can configure Integration Server to monitor a connection factory used by an active connection by polling the JNDI provider for changes. In this situation, Integration Server periodically compares

the connection factory used to create the connection with the connection factory in the JNDI provider. If Integration Server detects a change, it refreshes the connection using the new connection factory definition automatically.

Two factors determine how frequently Integration Server polls the JNDI provider for changes to a connection factory object:

- The polling interval specified in the JMS connection alias. You set the polling interval when you create or edit the JMS connection alias. The minimum polling interval that you can specify is 1 minute. Alternatively, you can instruct Integration Server to use the refresh interval specified for the webMethods connection factory object.
- The connection monitoring period, which determines how frequently, Integration Server checks the status of active connections to the webMethods Broker. The connection monitoring period is determined by the `watt.server.jms.connection.monitorPeriod` configuration property.

When the connection monitoring period elapses, Integration Server checks the status of the JMS connection alias. Integration Server also compares the polling interval to the time elapsed since the last poll to determine whether to poll for changes to the cluster connection factory.

To ensure that Integration Server polls for connection factory changes at the specified interval, the polling interval value should be greater than or equal to the value of the `watt.server.jms.connection.monitorPeriod` property. If the value of the polling interval is less than the value of the `watt.server.jms.connection.monitorPeriod` property, Integration Server will not check for connection factory changes at the specified interval.

For example, suppose that a JMS connection alias specifies a polling interval of 1 minute and `watt.server.jms.connection.monitorPeriod` is set to 10 minutes. Because checking the polling interval and, if necessary, polling for changes, occurs only when Integration Server checks the connection, Integration Server polls the cluster connection factory for changes every 10 minutes.

Note:

If the connection between Integration Server and the webMethods Broker fails, Integration Server attempts to reconnect to the webMethods Broker at an interval determined by the `watt.server.jms.connection.retryPeriod` property. After Integration Server restores the connection, Integration Server polls immediately for changes to the cluster connection factory specified in the JMS connection alias.

Registering an Event Listener

You can monitor a connection factory used by a JMS connection alias for changes by registering an event listener with the JNDI provider. Specifically, Integration Server registers an event listener to be notified about changes to the connection factory object in the JNDI namespace. After registering an event listener, Integration Server can receive notifications for change events or error events for the cluster connection factory object.

Note: Integration Server can register an event listener with any JNDI provider that supports EventContext interface. The webMethods JNDI provider supports the EventContext interface.

A *change event* indicates that changes occurred in the connection factory object used by the JMS connection alias. When Integration Server receives a change event for the connection factory object,

Integration Server refreshes the connection to the webMethods Broker using the new cluster connection factory object.

An *error event* indicates that a NamingExceptionEvent occurred on the JNDI provider. A NamingExceptionEvent can occur for multiple reasons, one of which is if the connection between Integration Server and the JNDI provider fails. After a NamingExceptionEvent occurs, the JNDI provider deregisters the event listener. As part of monitoring the state of the connection to the webMethods Broker, Integration Server monitors the state of an event listener. If Integration Server determines that the event listener was deregistered, Integration Server re-registers the change listener, polls the connection factory object for changes, and, if necessary, refreshes the connection.

Note:

The value of the `watt.server.jms.connection.monitorPeriod` configuration property determines the frequency with which Integration Server checks the state of an active connection and any registered change listeners. The default value is 45 seconds.

How Integration Server Updates the Connection

After Integration Server determines that the cluster connection factory changed, Integration Server suspends all activity associated with the connection established for the associated JMS connection alias. Then Integration Server updates the connection. To ensure that in-flight messages are not lost and that duplicate messages are not introduced, Integration Server completes the following sequence of tasks as part of updating the connection:

1. Suspends any JMS triggers that use the connection that needs to be restarted. The JMS trigger does not retrieve any more messages from the webMethods Broker, but continues processing any messages it has already retrieved. After the JMS trigger processes all the retrieved messages, Integration Server disables the JMS trigger.
2. Temporarily blocks any new threads for services that send a JMS message using the JMS connection alias for the connection that needs to be restarted, specifically `pub.jms:send`, `pub.jms:sendAndWait`, and `pub.jms:reply`.

The value of the `watt.server.jms.connection.update.blockingTime` server configuration parameter specifies the maximum amount of time the `pub.jms*` services will wait for a connection while the connection used by the service is being updated. The default is 1000 milliseconds. If Integration Server does not restart the connection before the blocking time elapses, Integration Server throws an `ISRuntimeException`.

3. Waits for any `pub.jms*` services that are in the midst of using the connection to send JMS messages to execute to completion.

The value of the `watt.server.jms.connection.update.restartDelay` server configuration parameter determines how long Integration Server waits for services sending JMS messages to execute to completion before restarting the connection. If the restart delay elapses before the `pub.jms*` services finish executing, Integration Server throws an `ISRuntimeException`.

4. Stops the connection for the JMS connection alias.
5. Refreshes the connection using the new cluster connection factory object. Refreshing the connection effectively enables the JMS triggers.

6. Releases any blocking threads for pub.jms* services.

Configuring Integration Server to Monitor a Connection Factory Object

Keep the following points in mind when configuring Integration Server to monitor the connection factory object used by a JMS connection alias.

- You can only monitor webMethods Connection Factory objects in a JNDI server.
- When monitoring a composite cluster connection factory object, Integration Server monitors for changes in the composite cluster connection factory object only. Integration Server does not monitor the constituent cluster connection factory objects for changes.
- To use a change listener to monitor the connection factory, the JNDI provider must support the EventContext interface.
- You can configure Integration Server to use the **Poll for changes (interval defined by webMethods Connection Factory)** option when monitoring a cluster or composite connection factory only.
- When monitoring a connection factory by polling for changes, the frequency with which Integration Server polls for changes is dependent upon the connection monitoring period. Integration Server checks the connection at frequency determined by the connection monitoring period. As part of checking the connection, Integration Server determines whether the polling interval has elapsed. If it has, Integration Server polls for changes to the cluster connection factory. If the polling interval is less than the connection monitoring period (controlled by `watt.server.jms.connection.monitorPeriod` parameter), Integration Server does not poll for changes with the frequency specified by the polling interval.
- You must disable a JMS connection alias before you can edit it.

Note:

The procedure below provides instructions for configuring monitoring for a JMS connection alias that already exists. You can also configure monitoring at the time you create an alias.

➤ **To configure Integration Server to monitor a connection factory**

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. Under **JMS Connection Alias Definitions**, select the JMS connection alias for which you want to monitor the associated connection factory for changes.
5. Click **Edit JMS Connection Alias**.

6. Under Connection Protocol Settings, in the **Monitor webMethods Connection Factory** list, select one of the following:

Select	To
No	Indicate that Integration Server will not monitor the connection factory. This is the default.
Poll for changes (specify interval)	Monitor the connection factory by polling for changes at an interval that you specify.
Poll for changes (interval defined by webMethods Connection Factory)	Monitor the connection factory at an interval determined by the refresh interval specified for the webMethods Connection Factory object. For more information about configuring a cluster connection factory, see <i>Administering webMethods Broker</i> and <i>webMethods Broker Messaging Programmer's Guide</i> .
Register change listener	Monitor the connection factory by registering an event listener.

7. If you selected **Poll for changes (specify interval)**, in the **Polling Interval (minutes)**, specify the number of minutes between polling attempts. The polling interval must be a positive integer. The default value is 60 minutes.
8. Click **Save Changes**.

Notes

When Integration Server starts the JMS connection alias, Integration Server does the following:

- Verifies that the JMS connection alias connects to the webMethods Broker
- Verifies that the JNDI provider supports the EventContext interface, if the JMS connection alias monitors the connection factory via an event listener

Integration Server throws a `JMSSubsystemException` if the any of the above criteria are not met.

Using SSL with JMS

Integration Server can send and receive messages to a JMS provider in a secure manner using SSL. If the connection between Integration Server and the JMS provider is to use SSL, you must configure SSL capabilities on the JMS provider and Integration Server. For requirements and information about configuring SSL on the JMS provider, consult the JMS provider documentation.

To configure SSL on Integration Server (the JMS client), you must provide certificates that can be used during the SSL handshake. Because certificates are stored in keystores and truststores, part of configuring Integration Server for using SSL is specifying the location of the keystores and truststore to be used for establishing the SSL connection. You can accomplish this a few different ways in Integration Server, specifically:

- Set the `javax.net.ssl` properties in the Integration Server JVM to provide certificate location and password information. When connecting to the JMS provider, Integration Server uses the `javax.net.ssl` property values for creating the SSL context of the SSL handshake.

While you can set the keystore location, truststore location, and password information directly in the `javax.net.ssl` properties for the JVM, this approach represents a security vulnerability because the properties take String values. Storing keystore and password information in plain text on the file system is not secure. If you do not want to store certificate location and password information in plain text, you can assign a keystore alias and/or truststore alias to be used by the JVM when establishing the default SSL context. At start up, Integration Server sets the `javax.net.ssl` properties by obtaining the store locations and passwords from the aliases and then creates the default SSL context. For more information, see [“Storing SSL Information for the Integration Server JVM in a Secure Manner” on page 464](#).

If you use the `javax.net.ssl` properties to provide the certificate and password information, you do not need to specify SSL configuration information in the JMS connection alias or for the initial JNDI context in the JNDI provider alias. However, using the `javax.net.ssl` properties for establishing an SSL connection with the JMS provider limits Integration Server to using one set of certificates for all SSL connections.

Some JMS providers require that JMS clients use the JVM default SSL context for the SSL handshake. Review your JMS provider documentation for requirements.

- Set JMS provider-specific SSL properties in the JNDI provider alias used with the JMS connection alias. Some JMS providers use a set of properties (key-value pairs) to supply certificate details. You can add these properties to a JNDI provider alias. Integration Server adds the certificate information to the JNDI context when the context is created. Supplying certificate and password information via the JNDI provider alias makes it possible to use multiple certificates with the JMS provider and allows the use of a different set of certificates than the ones configured for the JVM. This approach is applicable when the JNDI provider and JMS provider use the same set of certificates such as when using Universal Messaging as the JMS and JNDI providers.

For more information about supplying certificate details via the JNDI provider alias, see [“Including SSL Configuration Information in the JNDI Provider Alias” on page 281](#).

- When webMethods Broker is the JMS provider and using the native webMethods API to connect to the webMethods Broker, set the SSL information (Keystore, Keystore Type, Truststore, and Truststore Type) when you create the JMS connection alias.

Note:

If Integration Server connects to the webMethods Broker using JNDI, you need to configure the connection factory on the webMethods Broker. For more information, see *Administering webMethods Broker*.

Supported JMS Providers

Integration Server is certified to use with the following JMS providers:

JMS Provider	Version(s)
AMQP with messaging providers RabbitMQ and Azure Service Bus	AMQP 0-8, 0-9, 0-9-1, and 0-10
Apache ActiveMQ Classic	5.16.*
IBM MQ	9.0
JBoss Messaging	1.4.0 SP3 and higher, up to but not including 2.0
Oracle Streams Advanced Queuing (AQ)	10.2.*, 11.2.0, and 12.2.0
Software AG Universal Messaging	10.5 and higher
Solace PubSub+	10.6.0
SonicMQ	7.5 and 7.6
WebLogic	9.1, 9.2, 10.3, and 12.1.3
webMethods Broker	6.5 and higher webMethods Broker is deprecated.
WebSphere Application Server	8.5

Considerations for JMS Providers

Ensure to review your JMS provider documentation for information about any specific considerations for JMS support.

AMQP Messaging Providers

Integration Server is certified to communicate with the RabbitMQ and Azure Service Bus providers. Integration Server uses the QPID JMS client libraries for AMQP to communicate with these JMS providers. Integration Server was tested with the Qpid JMS AMQP 0-x 6.3.3 JMS client libraries.

Apache ActiveMQ

If you are using SSL/TLS to connect to Apache ActiveMQ broker, do the following:

- Use the keytool provided by Apache ActiveMQ to create a certificate for the broker, set up key and truststores, and import the ActiveMQ broker's public certificate into Integration Server.
- Use the `javax.net.ssl` properties to set the location of the keystore and truststore files for connecting with the ActiveMQ broker. For Integration Server, set the `javax.net.ssl` properties in the `custom_wrapper.conf` file. For example:
 - `wrapper.java.additional.n= -Djavax.net.ssl.keystore=keystoreLocation`

- `wrapper.java.additional.n+1=-Djavax.net.ssl.keyStorePassword=password`
- `wrapper.java.additional.n+2=-Djavax.net.ssl.trustStore=truststoreLocation`

For Microservices Runtime, set Java system properties in the following file using the `JAVA_CUSTOM_OPTS` property: *Integration Server_directory* /bin/setenv.bat(sh) .

- If you need to configure two-way SSL with the ActiveMQ broker, import the public certificate for Integration Server into the ActiveMQ broker truststore.

IBM MQ

Integration Server is certified to communicate with IBM MQ 9.0 JMS provider. Consider the following points while connecting to IBM MQ 9.0.

- If you are setting up an SSL connection to IBM MQ 9.0, consider modifying the CipherSuite value in the JNDI `.bindings` file based on the JRE specified for Integration Server and IBM's recommendation.
- By default, IBM MQ 9.0 supports JMS 2.0 specification. However, Integration Server supports JMS 1.1 specification. Therefore, set the following property to `true` in *Software AG_directory* \ profiles\IS_instance_name\configuration\custom_wrapper.conf file so that Integration Server functions seamlessly with IBM MQ 9.0.

```
wrapper.java.additional.availableNumber=-Dcom.ibm.mq.jms.SupportMQExtensions=true
```

Alternatively, you can set the value of `watt.config.systemProperties` property as `com.ibm.mq.jms.SupportMQExtensions=true`.

For Microservices Runtime, set Java system properties in the following file using the `JAVA_CUSTOM_OPTS` property: *Integration Server_directory* /bin/setenv.bat(sh) .

Solace PubSub+ Appliance and Cloud

Direct Transport mode in Solace PubSub+ does not support the following features:

- Message selector
- Local transaction support
- Durable subscription

Note: *Guaranteed Transport mode* in Solace PubSub+ supports the above features.

If you use *Direct Transport mode* in Solace PubSub+ and your JMS provider uses any of these features, Integration Server responds in these ways:

- If a JMS trigger is subscribed to a topic and uses the *Message selector* feature, then Integration Server does not enable the trigger and displays the message, Error creating consumer - cannot use selector with direct messaging.

- If the transaction type of a JMS connection alias pointing to Solace PubSub+ is *Local transaction* or *XA transaction*, then Integration Server cannot execute the JMS send and receive operations and displays the message, Transacted sessions or XA sessions are not supported with direct transport.
- If a JMS trigger is subscribed to a topic with durable subscriber name, then Integration Server does not enable the trigger and displays the message, Unknown Durable Topic Endpoint.

About Using Software AG Universal Messaging as the JMS Provider

Keep the following points in mind when using Universal Messaging as the JMS provider:

- When using Universal Messaging, if the version of Universal Messaging is equivalent to or higher than the version of Integration Server, you do not need to add any client libraries to the Integration Server classpath.
- Keep the Universal Messaging client libraries up to date. If you install a Universal Messaging fix that updates the client libraries, make sure to copy the updated Universal Messaging client library files to the *Software AG_directory* /common/lib directory used by Integration Server.
- When using Integration Server with a higher version of Universal Messaging, Integration Server cannot use any features delivered in the higher version of Universal Messaging. Integration Server can use only the features that existed in the equivalent version of Universal Messaging.
- Integration Server uses synchronous publishing as the default when publishing persistent messages to Universal Messaging.

When using Universal Messaging as the JMS provider, the JMS client can use synchronous or asynchronous publishing. With synchronous publishing, the JMS client sends messages to Universal Messaging one at a time. The JMS client receives a response indicating a successful send only after the message has been delivered to Universal Messaging. With asynchronous publishing, the JMS messages are placed in a buffer and then sent to Universal Messaging in batches. However, the JMS client can receive a response indicating a successful send before the message is actually delivered to Universal Messaging. Asynchronous publishing is faster than synchronous publishing, but messages can be lost in the event the connection to Universal Messaging fails or Integration Server becomes unavailable. The default for a JMS client using Universal Messaging as the JMS provider is asynchronous publishing. This default applies to all messages regardless of a persistent or non-persistent delivery mode. However, to ensure delivery of a persistent message, Integration Server always uses synchronous publishing to send a persistent JMS message to Universal Messaging.

- Message priority is not supported when Universal Messaging is the JMS provider. Any priority assigned to a JMS message sent to Universal Messaging will be ignored.
- If you are using Universal Messaging as the JMS provider and you want to use durable subscribers, ensure to select the **Shared Durable** option when creating a `ConnectionFactory` or `TopicConnectionFactory` using Universal Messaging Enterprise Manager. For example, when creating a `ConnectionFactory`, select the option named **Connection Factory (Shared Durable)**.

- When an Integration Server JMS connection alias is connected to a cluster of Universal Messaging realm servers and the master realm server in the cluster goes down, Integration Server stops the alias. This causes all JMS triggers associated with the alias to stop, and all JMS sending services to fail with a `ResourceUnavailableException`. The alias automatically reconnects to the cluster if and when the cluster is able to reconcile itself (that is, the cluster is able to reelect a new master realm server).
- If Integration Server publishes documents to a Universal Messaging server on which the `PauseServerPublishing` configuration parameter is set to `true`, publishing fails with an `nPublishPauseException`. Integration Server handles the `nPublishPauseException` by proceeding as if the messaging connection alias is unavailable.
- Integration Server enforces minimum values for the server configuration properties that control the polling intervals for JMS triggers when Universal Messaging is the JMS provider. The default values for these parameters are not optimal when connecting to Universal Messaging. Lower polling intervals, including the default values, can result in high CPU utilization on the Universal Messaging server and do not provide a performance benefit when working with Universal Messaging.

The following table identifies the minimum values for the server configuration parameters that control polling intervals when Universal Messaging is the JMS provider.

Parameter	Minimum value used with Universal Messaging
<code>watt.server.jms.trigger.concurrent.primaryThread.pollingInterval</code>	3000 milliseconds
<code>watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval</code>	3000 milliseconds if the concurrent JMS trigger does not use a join 10 milliseconds if the concurrent JMS trigger uses a join
<code>watt.server.jms.trigger.serial.primaryThread.pollingInterval</code>	3000 milliseconds

Note:

If the serial JMS trigger uses a join, Integration Server uses a secondary polling interval which is set to 10 milliseconds. Note that the secondary polling interval is not configurable for a serial JMS trigger.

You can increase the polling intervals by changing the server configuration parameters. Keep in mind that the values you specify will be used with all JMS providers. Integration Server only enforces a minimum when connecting to Universal Messaging.

- JMS triggers that use joins might affect the CPU utilization for Universal Messaging even with the minimum values mentioned above. This is because the secondary polling interval is low (10 milliseconds). The secondary polling interval needs to be low so that the JMS trigger can

quickly retrieve messages from the different destinations. To reduce the impact on Universal Messaging, consider one of the following approaches:

- For a concurrent JMS trigger, configure the JMS connection alias used by the trigger to use multiple connections and then set the JMS trigger **Connection count** property to a value that is evenly divisible by the number of destinations. For example, if a JMS trigger retrieves messages from two destinations, set **Connection count** to 2, 4, 6, 8, and so on. If the JMS trigger retrieves messages from three destinations, set **Connection count** to 3, 6, 9, and so on.
- For a serial or concurrent JMS trigger, instead of using a join for the JMS trigger, use a Universal Messaging channel join to gather messages from different destinations into a single destination on the Universal Messaging server.
- A Universal Messaging client, such as Integration Server caches some information about destination objects. Integration Server, keeps this information in the client store cache. However, when particular queue or channel properties (such as Time to Live and Maximum Number of Events) are changed using Universal Messaging Enterprise Manager, Universal Messaging server creates a new destination object for the channel or queue. However, Integration Server is not aware of the new destination object for the channel or queue and continues to use the obsolete cached destination object. Consequently, sending messages to the channel or queue fails because Integration Server cannot locate the destination. JMS triggers cannot receive messages from the channel or queue fails for the same reason. To resolve this issue and allow Integration Server to connect to the destination, you must restart the JMS connection alias by first disabling the alias and then enabling it.

Adding JMS Provider Client Libraries to Integration Server Classpath

Integration Server requires access to some of the JMS provider's Java client libraries. You can do one of the following:

- Place the libraries in the server's classpath by placing them in the *Integration Server_directory* \instances\ *instance_name* \lib\jars\custom directory, as described in the procedure below.
- Make the libraries available to all server instances by placing them in the *Integration Server_directory* \lib\jars\custom directory.

Note:

Files located in the server's classpath take precedence over the files located at the *Integration Server_directory* \lib\jars\custom directory.

- Isolate the jars within a package classloader by placing them in the following directory: *packageName* \code\jars.

If you place the files in the package classloader, make sure to set the **Class Loader** property when configuring a JMS connection alias to this JMS provider.

The JMS provider's .jar files could potentially conflict with other webMethods components running on Integration Server. If you encounter a conflict, consider using a package class loader instead.

Note:

The webMethods Broker and JNDI provider .jar files are already included in Integration Server. The Universal Messaging .jar files are included in *Software AG_directory /common/lib*.

➤ **To add a third party JMS provider's JMS and JNDI libraries to the Integration Server classpath:**

1. Copy your JMS provider's Java API libraries from the locations specified below and place them in *Integration Server_directory \instances \instance_name \lib \jars \custom* directory.

JMS Provider	Files to Copy
AMQP message formats	<p>From Apache QPID library:</p> <p>apache-qp-id-jms-amqp-0-x-6.3.3-bin.tar.gz</p> <p>OR</p> <p>apache-qp-id-jms-amqp-0-x-6.3.3-bin.zip</p>
Apache ActiveMQ Classic	<ul style="list-style-type: none"> ■ <i>ActiveMQ_HOME</i> \activemq-all-5.16.*.jar <p>Where <i>ActiveMQ_HOME</i> is the directory in which Apache ActiveMQ is installed, for example: <i>apache-activemq-5.16.1-bin \apache-activemq-5.16.1</i></p>
IBM MQ	<p>For IBM MQ version 9.0:</p> <ul style="list-style-type: none"> ■ <i>IBMMQ</i> \Java \lib \com.ibm.mq.allclient.jar ■ <i>IBMMQ</i> \Java \lib \fscontext.jar ■ <i>IBMMQ</i> \Java \lib \providerutil.jar ■ <i>IBMMQ</i> \Java \lib \jms.jar <p>Where <i>IBMMQ</i> is the directory in which your IBM MQ is installed.</p>
JBoss Messaging	<ul style="list-style-type: none"> ■ jboss-messaging-client.jar (This is available in the JBoss Messaging distribution.) ■ <i>JBOSS_HOME</i> \client \jbossall-client.jar ■ <i>JBOSS_HOME</i> \server \SERVER_NAME \deploy \jboss-aop-deployer \jboss-aop.jar ■ <i>JBOSS_HOME</i> \server \SERVER_NAME \lib \javassist.jar ■ <i>JBOSS_HOME</i> \server \SERVER_NAME \lib \trove.jar <p>Where <i>JBOSS_HOME</i> is the directory in which JBoss is installed and <i>SERVER_NAME</i> is the name of the messaging server.</p>

Note:

JMS Provider	Files to Copy
	<p>JBoss Messaging 1.4.0 requires a patched version of <code>jboss-remoting.jar</code>.</p>
Oracle Streams Advanced Queuing (AQ)	<p>For Oracle Streams Advanced Queuing (AQ) 10.2.*:</p> <ul style="list-style-type: none"> ■ <code>ORACLE_HOME\jdbc\lib\classes12.jar</code> ■ <code>ORACLE_HOME\jlib\orai18n.jar</code> ■ <code>ORACLE_HOME\lib\xmlparserv2.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\xdb.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\aqapi13.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\jmscommon.jar</code>
	<p>Where <code>ORACLE_HOME</code> is <code>ORACLE_BASE\product\10.2.0\db_1</code> and <code>ORACLE_BASE</code> is the directory in which the Oracle database is installed.</p>
	<p>For Oracle Streams Advanced Queuing (AQ) 11.2.0:</p> <ul style="list-style-type: none"> ■ <code>ORACLE_HOME\jlib\orai18n.jar</code> ■ <code>ORACLE_HOME\lib\xmlparserv2.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\xdb.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\aqapi.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\jmscommon.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\ojdbc6.jar</code>
	<p>Where <code>ORACLE_HOME</code> is <code>ORACLE_BASE\product\11.2.0\db_1</code> and <code>ORACLE_BASE</code> is the directory in which the Oracle database is installed.</p>
	<p>Note: If you are using Oracle Streams Advanced Queuing (AQ) 11.2.0 as the JMS provider, the value of the <code>watt.config.systemProperties</code> property must include <code>"oracle.jms.conservativeNavigation=true"</code>.</p>
	<p>For Oracle Streams Advanced Queuing (AQ) 12.2.0:</p> <ul style="list-style-type: none"> ■ <code>ORACLE_HOME\jdbc\lib\ojdbc8.jar</code> ■ <code>ORACLE_HOME\jlib\orai18n.jar</code> ■ <code>ORACLE_HOME\lib\xmlparserv2.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\aqapi.jar</code>

JMS Provider	Files to Copy
	<ul style="list-style-type: none"> ■ <code>ORACLE_HOME\rdbms\jlib\jmscommon.jar</code> ■ <code>ORACLE_HOME\rdbms\jlib\xdb.jar</code> <p>Where <code>ORACLE_HOME</code> is <code>ORACLE_BASE\product\12.2.0\db_1</code> and <code>ORACLE_BASE</code> is the directory in which the Oracle database is installed.</p>
Solace PubSub+	<p>From Solace library or from Solace installation directory:</p> <ul style="list-style-type: none"> ■ <code>sol-common-10.6.0.jar</code> ■ <code>sol-jcsmp-10.6.0.jar</code> ■ <code>sol-jms-10.6.0.jar</code>
SonicMQ	<p>For SonicMQ version 7.5:</p> <ul style="list-style-type: none"> ■ <code>SonicMQ_directory\lib\mfcontext.jar</code> ■ <code>SonicMQ_directory\lib\sonic_Client.jar</code> ■ <code>SonicMQ_directory\lib\sonic_Crypto.jar</code> ■ <code>SonicMQ_directory\lib\sonic_mgmt_client.jar</code> ■ <code>SonicMQ_directory\lib\sonic_Selector.jar</code> ■ <code>SonicMQ_directory\lib\sonic_XA.jar</code> <p>For SonicMQ version 7.6, all of the above, plus:</p> <ul style="list-style-type: none"> ■ <code>SonicMQ_directory\lib\mgmt_client.jar</code> ■ <code>SonicMQ_directory\lib\mgmt_config.jar</code> ■ <code>SonicMQ_directory\lib\sonic_XMessage.jar</code> <p>Where <code>SonicMQ_directory</code> is the directory in which SonicMQ is installed.</p>
WebLogic	<p>For WebLogic 9.1 and 9.2:</p> <ul style="list-style-type: none"> ■ <code>WebLogic_directory\server\lib\weblogic.jar</code> <p>For WebLogic 10.3 and 12.1.3:</p> <ul style="list-style-type: none"> ■ <code>WebLogic_directory\server\lib\wljmsclient.jar</code> ■ <code>WebLogic_directory\server\lib\wlnmclient.jar</code> ■ <code>WebLogic_directory\server\lib\wlclient.jar</code>

JMS Provider	Files to Copy
WebSphere Application Server	<p>Where <i>WebLogic_directory</i> is the directory in which WebLogic is installed.</p> <p>For WebSphere Application Server version 8.5:</p> <ul style="list-style-type: none"> ■ <i>WebSphereApplicationServer</i>\AppServer\runtimes\com.ibm.wsslbndint_850.jar ■ <i>WebSphereApplicationServer</i>\AppServer\runtimes\com.ibm.ws.orb_8.5.0.jar ■ <i>WebSphereApplicationServer</i>\AppServer\runtimes\com.ibm.wsslbndintjms_850.jar <p>Where <i>WebSphereApplicationServer</i> is the directory in which your WebSphere Application Server is installed.</p>

2. Restart Integration Server.

Note:

The list of files for each JMS provider is a general guideline. The requirements for each JMS provider may change. Ensure that you review the documentation for your JMS provider to determine an exact list of required jar files. Furthermore, review your chosen JMS provider documentation to determine how to handle fixes and updates to the JMS provider's Java client libraries. Updates or fixes to the Java client libraries may require updates to the provider client libraries added to the Integration Server classpath.

13 Configuring Integration Server for MQTT Messaging

■ Overview of MQTT Support in Integration Server	326
■ Supported MQTT Servers	327
■ Creating an MQTT Connection Alias	328
■ Editing an MQTT Connection Alias	331
■ Enabling and Disabling an MQTT Connection Alias	332
■ Deleting an MQTT Connection Alias	333
■ Specifying a Retry Interval for Failed Connections to the MQTT Server	333

Overview of MQTT Support in Integration Server

MQTT is a publish-and-subscribe messaging protocol often used with devices in the Internet of Things (IoT) and machine-to-machine (M2M) communication. MQTT was designed to be light-weight and used with devices for which resources and network bandwidth are constrained, such as a sensor on a smart device. The protocol supports telemetry applications which measure and transmit data from sensors and devices.

Note:

A full discussion of the capabilities of MQTT is beyond the scope of this document but is available elsewhere.

Like other publish-and-subscribe solutions, MQTT consists of clients that publish or subscribe to messages and a server that routes published messages to subscribers.

Integration Server, which supports MQTT version 3.1.1, can act as an MQTT client that publishes messages to a topic on an MQTT server and an MQTT client that subscribes to topics on the MQTT server. Specifically:

- Integration Server can publish MQTT messages to an MQTT server using the built-in service `pub.mqtt:publish`.
- Integration Server can subscribe to topics by creating an MQTT trigger. The MQTT trigger receives messages published to the topic on the MQTT server and then invokes a trigger service to process the messages.
- Integration Server uses an MQTT connection alias to create a connection to the MQTT server. An invocation of the `pub.mqtt:publish` service requires the MQTT connection alias to publish the message to the MQTT server. Similarly, an MQTT trigger specifies the MQTT connection alias that it uses to identify the MQTT server from which it retrieves messages and on which the trigger creates subscriptions.

Note:

An MQTT server is also referred to as an MQTT broker. The Integration Server documentation uses the term MQTT server to avoid confusion with the webMethods Broker product.

Adding an MQTT solution to Integration Server consists of the basic tasks listed in the following table.

- | | |
|----------------|--|
| Stage 1 | Determine which MQTT provider you will use. For a list of supported MQTT providers, see “Supported MQTT Servers” on page 327 . |
| Stage 2 | Create one or more MQTT connection aliases. For information about creating an MQTT connection alias, see “Creating an MQTT Connection Alias” on page 328 . |
| Stage 3 | Build MQTT triggers. This step is necessary only if you intend to use Integration Server to retrieve and process MQTT messages from the MQTT server. |

For information about building MQTT triggers, see *webMethods Service Development Help*.

- Stage 4** Build services that publish MQTT messages using the `pub.mqtt:publishservice`. This step is necessary only if you want to use Integration Server to publish MQTT messages. For more information about the `pub.mqtt:publishservice`, see the *webMethods Integration Server Built-In Services Reference*.

Limitations of MQTT Support in Integration Server

MQTT support in Integration Server has the following limitations:

- Integration Server does not support SSL for connections to the MQTT server. Integration Server supports basic authentication only.
- Enhanced logging, which instructs Integration Server to write detailed log entries when sending, receiving or processing messages, cannot be used with MQTT.
- MQTT connection aliases do not have a client side queue (CSQ) which is an available feature for other types of messaging connection aliases in Integration Server.
- Exactly-once message processing feature is not available for MQTT triggers.
- In a cluster of Integration Servers, identically named MQTT triggers and MQTT connection aliases should not exist on each node in the cluster. Furthermore, MQTT connection aliases in a cluster should all use unique connection client IDs. No duplicate connection client IDs should be used.

The cluster limitations are because the MQTT specification does not support multiple clients sharing a subscription. Identical MQTT triggers share the same subscriptions and the same connection client ID. The connection client ID of the MQTT trigger is the connection client ID value of the MQTT connection alias plus the trigger name.

Because of adherence to the MQTT specification, some MQTT providers do not support a shared subscription and the use of the same connection client ID by multiple clients. Some MQTT providers may offer a workaround for this restriction.

Supported MQTT Servers

Integration Server supports MQTT version 3.1.1. Integration Server is certified to work with the MQTT servers listed in the following table.

MQTT Server	Version
Eclipse Mosquitto	1.5 and 1.6
Universal Messaging	10.5

The Eclipse Paho libraries version 1.2.1, which includes client implementations and utilities for the MQTT messaging protocol, are installed with Integration Server in *Software AG_directory/common/lib*. The Paho libraries can be used to connect to Universal Messaging and Mosquitto.

Creating an MQTT Connection Alias

An MQTT connection alias contains the configuration information that Integration Server needs to establish a connection to an MQTT server. Integration Server uses an MQTT connection alias when publishing an MQTT message. An MQTT trigger uses an MQTT connection alias to retrieve messages from the topics to which it subscribes on the MQTT server. Each Integration Server can have multiple MQTT connection aliases.

Use the following procedure to create an MQTT connection alias.

➤ To create an MQTT connection alias

1. Open Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under MQTT Configuration, click **MQTT Settings**.
4. Click **Create MQTT Connection Alias**.
5. Under **General Settings**, specify values for the fields in the following table:

Field	Description
Connection Alias Name	Unique name for the MQTT connection alias. Each MQTT connection alias contains the configuration information needed for Integration Server to establish a connection to the MQTT server.
Description	Description for the MQTT connection alias.
Package	Name of the package that you want to associate with the MQTT connection alias. Integration Server saves the connection alias configuration in this package. When you assign an MQTT connection alias to a package, Integration Server enables and disables the alias in conjunction with the package. If you disable the package, Integration Server disables the MQTT connection alias automatically. A package retains its MQTT connection alias when the package is published to other servers. If a subscriber installs the package, the MQTT connection alias is created automatically on the subscriber's Integration Server. If you do not have a package to assign to the MQTT connection alias, leave the Package setting at WmRoot (the server's

Field	Description
	core package). This makes the MQTT connection alias available whenever Integration Server is running

6. Under **Connection Protocol Settings**, specify values for the fields in the following table

Field	Description
Host	URL containing the protocol, domain, and port of the MQTT server.
Connection Client ID	Client identifier for the connections associated with this MQTT connection alias. Integration Server generates a connection client identifier automatically, but you can supply your own
Use Clean Session for Subscriptions	<p>Whether connections created for MQTT triggers using this MQTT connection alias establish a persistent session on the MQTT server. Do one of the following:</p> <ul style="list-style-type: none"> ■ Select the Use Clean Session for Subscriptions check box to remove any previous subscriptions established by an MQTT trigger that uses this MQTT connection alias. When the session ends, the MQTT server will remove any subscriptions created by the MQTT trigger. ■ Clear the Use Clean Session for Subscriptions check box to persist subscriptions across sessions. Any new subscriptions created by an MQTT trigger during a session will be added to the existing subscriptions for the trigger. Additionally, the subscriptions remain after the trigger disconnects. Messages sent while the trigger is disconnected will be kept if the Quality of Service (QoS) is 1 or 2. The trigger will retrieve missed messages when it reconnects.
Connection Timeout (seconds)	Maximum number of seconds that Integration Server waits for a network connection to the MQTT server to be established. A value of 0 disables the timeout, which means that Integration Server waits for a failed or successful network connection attempt. The default is 30 seconds.
Keep Alive (seconds)	The maximum number of seconds that can elapse between messages being sent to or received from the MQTT server using a connection created for this MQTT connection alias. The keep alive functionality in MQTT

Field	Description
	ensures that the connection between the MQTT client (IS) and the MQTT server is open. A value of 0 disables keep alive. The default is 60 seconds.

7. Under **Last-Will Settings**, enter the information you want Integration Server to provide in the event of an ungraceful disconnection from the MQTT server.

Field	Description
Enable Last-Will	<p>Whether the MQTT server sends a will message on behalf of Integration Server if Integration Server disconnects abruptly from the MQTT server. A will message allows other MQTT clients to handle an abrupt disconnection. Select Enable Last-Will Message if you want the MQTT server to send a specific message to a particular topic when Integration Server disconnect abruptly from the MQTT server. Do one of the following:</p> <ul style="list-style-type: none">■ Select the Enable Last-Will check box to send a will message if Integration Server disconnects ungracefully.■ Clear the Enable Last-Will check box if Integration Server will not send a will message upon ungraceful disconnection.
Last-Will QoS	<p>The Quality of Service for the will message. Select one of the following:</p> <ul style="list-style-type: none">■ At most once (0)■ At least once (1)■ Exactly once (2)
Last-Will Retained	<p>Whether the MQTT server retains the last will message. Do one of the following:</p> <ul style="list-style-type: none">■ Select the Last-Will Message Retained check box to instruct the MQTT server to retain the last will message.■ Clear the Last-Will Message Retained check box if the MQTT server will not retain the last will message.
Last-Will Topic	<p>The topic to which to publish the last will message. All subscribers to this topic will receive the last will message. The topic does not need to exist at the time you create the MQTT connection alias. If it does not exist the first time</p>

Field	Description
	Integration Server uses the MQTT connection alias to establish connection, the MQTT server creates the topic.
Last-Will Message Payload	The contents of the last will message.
On Connect Message Payload	The message to send upon successful connection to the MQTT server. This is the payload of the connect message and is sent to the same topic as the Last-Will Message Topic .
On Disconnect Message Payload	The message to send upon disconnecting from the MQTT server. This is the payload of the disconnect message and is sent to the same topic as the Last-Will Message Topic .

- Under **Security Settings**, if basic authentication is required to connect to the MQTT server, provide the user name and password Integration Server uses.

Field	Description
User Name	User name for the Integration Server to provide for client authentication
Password	Password for the user name.

- Click **Save Changes**.

- Enable the MQTT connection alias.

Editing an MQTT Connection Alias

You might need to edit an MQTT connection alias. For example, you might want to change the alias so that subscribers do not use a clean session. Or, you might want to change an MQTT alias so that the MQTT server no longer sends a last will message on behalf of Integration Server when Integration Server disconnects abruptly.

Keep the following information in mind when you edit an MQTT connection alias.

- You must disable an MQTT connection alias before you can edit it.
- The MQTT connection alias name cannot be edited.
- Changing the **Connection Client ID** may result in new subscriptions on the MQTT server for MQTT triggers that use the alias. The subscriptions that use the old connection client ID are not automatically removed from the MQTT server. Old subscriptions might contain messages and continue to receive messages.

➤ To edit an MQTT connection alias

1. Open Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under MQTT Configuration, click **MQTT Settings**.
4. In the MQTT Connection Alias Definitions list, select the MQTT connection alias that you want to edit.
5. Click **Edit MQTT Connection Alias**.

If the **Edit MQTT Connection Alias** link is not available (it is not underlined), return to the MQTT Setting page and disable the alias.

6. Edit the properties of the MQTT connection alias.
7. Click **Save Changes**.
8. Enable the MQTT connection alias.

Enabling and Disabling an MQTT Connection Alias

When an MQTT connection alias is enabled, Integration Server can use the alias to publish messages to the MQTT server., create subscriptions to topics on behalf of MQTT triggers that use the alias, and retrieve messages from a topic to which an MQTT trigger subscribes. When an MQTT connection alias is disabled, Integration Server stops all MQTT triggers that use the alias. The trigger status will be Enabled (Connection alias not running). Additionally, when an alias is disabled, any services that use the MQTT connection alias to publish messages to an MQTT server will end with an error.

➤ To enable or disable an MQTT connection alias

1. Open Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under MQTT Configuration, click **MQTT Settings**.
4. In the MQTT Connection Alias Definitions list, do one of the following in the **Enabled** column:
 - Click **No** if the alias is disabled and you want to enable it.
 - Click **Yes** if the alias is enabled and you want to disable it.

Deleting an MQTT Connection Alias

Before you delete an MQTT connection alias, make sure of the following:

- The MQTT connection alias is disabled.
- A service that publishes MQTT messages does not rely on the MQTT connection alias.
- An MQTT trigger does not use the MQTT connection alias to connect to the MQTT server and retrieve message.

While you can delete an MQTT connection alias that is in use, it may result in error or failures in MQTT triggers and MQTT message publishing services.

➤ To delete an MQTT connection alias

1. Open Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under MQTT Configuration, click **MQTT Settings**.
4. In the MQTT Connection Alias Definitions list, disable the alias if it is not already disabled.
5. Click **X** in the **Delete** column for the alias.

If an MQTT trigger uses the MQTT connection alias, Integration Server prompts you to confirm deleting the alias.

Specifying a Retry Interval for Failed Connections to the MQTT Server

When a connection between Integration Server and the MQTT server fails, Integration Server attempts to re-establish the connection automatically after a 20 second delay. You can configure how long Integration Server waits between attempts to re-establish the connection by changing the value of the `watt.server.commonmessaging.connection.retryPeriod` parameter.

14 Using Enhanced Logging for Messaging

■ About Enhanced Logging for Messaging	336
■ What Is in Log Entries for Enhanced Logging?	336
■ Using the Messaging Log or the Server Log for Enhanced Logging	338
■ Configuring Enhanced Logging for Messaging	339
■ Viewing the Messaging Log	340

About Enhanced Logging for Messaging

Enhanced logging for messaging is a configurable feature that instructs Integration Server to write detailed log entries when sending or receiving and processing messages. The detailed logging, referred to as enhanced logging, makes it possible to track an individual message across your Integration Servers from the time the message is published through the time a trigger receives, processes, and acknowledges the message.

Each log message presents information about the message progress in a consistent way, making it possible to track the progress of the message through Integration Server. Most importantly, whenever possible, enhanced logging entries include a unique identifier for the message being published or consumed. Being able to track individual messages can help in debugging issues in your messaging solution.

Integration Server provides many configuration options for the enhanced messaging logging. These options can help focus the log entries on particular areas of your messaging implementation as well as limit the impact of enhanced logging on performance. For example, you configure enhanced logging for an individual messaging connection alias and not the entire Integration Server. For each alias you can specify whether Integration Server generates additional logging for all message consumers and/or producers or just specific message consumers and/or producers. You can also specify whether or not the enhanced messaging logging is verbose by setting a logging level.

Enhanced logging is available for JMS messaging and webMethods messaging only. It is not available for MQTT messaging.

Note:

Enhanced logging is for Integration Server activities only. Enhanced logging will not contain information about activities that occur on the messaging provider.

What Is in Log Entries for Enhanced Logging?

Enhanced logging entries include details about the message for which the log entry is being written in addition to the typical time stamp and log message text that appear in other log entries.

These message details can include a UUID, the message ID assigned by the provider, and the connection alias used to send or receive the messages. Each log entry includes the message details in the same order, which may be helpful when reviewing the log entries.

The message information that appears in the log entry depends on the point in publishing or subscribing at which Integration Server writes the log entry. For example, a log entry about sending a message would not include the provider ID for the message because the messaging provider has not yet received the message. A log entry about writing a message to the client side queue would not include a trigger name. A log entry for preparing to publish a document to Universal Messaging does not include a UUID if one was not specified in the document or the publishing service does not allow custom UUIDs because Integration Server has not yet generated the UUID.

Note:

A log entry includes a message ID or UUID whenever possible, which makes it easier to track a message as it is published and/or received and processed by Integration Server.

The following table identifies and describes the possible information in each enhanced logging entry for a message being sent or received:

Message property Description

UUID	<p>Universally unique identifier for the message being sent or received.</p> <ul style="list-style-type: none"> ■ In webMethods messaging, you can set a value for the <i>_env/uuid</i> field when publishing the document. You can only use a user-defined UUID for messages sent to Universal Messaging by the <i>pub.publish:publish</i> service. If you do not set a UUID value or if you are using a service for a user-defined UUID cannot be set, Integration Server assigns one during publication. ■ For JMS messaging, you can set a UUID in the <i>JMSMessage/properties/uuid</i> field when sending the JMS message. For JMS messaging, the <i>uuid</i> field is optional. Integration Server does not generate or assign a UUID if one is not specified.
Messaging connection alias	<p>Name of the messaging connection alias associated with the sent or received message.</p> <ul style="list-style-type: none"> ■ For webMethods messaging, this is a Universal Messaging connection alias. Enhanced logging cannot be used with webMethods Broker ■ For JMS messaging, this is a JMS connection alias.
Destination	<p>Destination to which the message was sent.</p> <ul style="list-style-type: none"> ■ For webMethods messaging, the destination is the name of the channel to which the message was sent. The name of the channel corresponds to the publishable document type for which Integration Server published an instance document. ■ For JMS messaging, the destination contains the JNDI lookup name for the queue or topic to which the JMS message was sent.
Message ID	<p>The ID assigned to the message by the messaging provider.</p> <ul style="list-style-type: none"> ■ For webMethods messaging, the message ID is the Universal Messaging event ID which Universal Messaging generates when it first receives the message. ■ For JMS messaging, the message ID is the <i>JMSMessageID</i> which is generated on behalf of the JMS provider.
Trigger name	<p>Name of the trigger that received the message.</p>

Enhanced logging entries written to the *server.log* are not identical to the enhanced logging entries written to the dedicated messaging log. The formats used by the *server.log* entries and by the messaging log entries are different. However the *server.log* entries and messaging log entries contain the same log message text and message properties. For more information about directing

enhanced logging to the server.log or the messaging log, see [“Using the Messaging Log or the Server Log for Enhanced Logging”](#) on page 338.

Logging Levels for Enhanced Messaging Logging

The number of log entries written for a published or received message as part of enhanced logging depends on the logging level. For log entries written to the messaging log, the logging level for the messaging audit logger determines the logging level. For messages written to the server.log, the logging level for the 0168 Messaging (Enhanced Logging) server log facility determines the logging level.

Note:

The logging level for the messaging audit logger can be different than the logging level for the 0168 Messaging (Enhanced Logging) server log facility for the server log.

To generate enhanced logging for messaging, the logging level, must be set to one of the levels listed in the following table:

At this level	Integration Server logs entries for
Info	Main sending and receiving actions such as success or failure of sending, receiving, and processing messages. A logging level of Info is considered standard logging for enhanced messaging logging.
Debug	Details of sending and receiving activities such as preprocessing and message acknowledgment. A logging level of Debug is considered verbose logging for enhanced messaging logging.
Trace	Further details of sending and receiving activities, such as the start and end of polling the messaging provider for additional messages.

Note:

Trace-level messages are provided for JMS messaging only.

Note:

Each logging level includes the indicated type of messages plus all messages from the levels above it. For example, the Trace level includes Debug and Info messages.

Using the Messaging Log or the Server Log for Enhanced Logging

When you configure a messaging connection alias to use enhanced logging, you can specify that Integration Server write the log entries to the server.log or the messaging log.

The messaging log is an audit log exclusively for enhanced logging entries written for sending and receiving of messages with Integration Server. The messaging audit log is formatted to align the log entries, including the log message text and message properties, into columns. This may make the log easier to use.

The server log is a journal log and contains log entries for all of the server log facilities. You can view the messaging log and server log in Integration Server Administrator or with text editors.

You can use the messaging log or the server log as the destination for enhanced log entries. However, you cannot set the same messaging connection alias to write to both locations.

Configuring Enhanced Logging for Messaging

Integration Server provides many configuration options for enhanced logging. The granularity of the options makes it possible for Integration Server to generate logging data more tailored to your needs. For example, perhaps you want to see logging for all messages sent using a particular messaging connection alias. Or maybe you want a narrower focus in the logging entries and only want to see log entries for messages sent by a messaging connection alias to a particular destination.

When configuring enhanced logging for messaging, you need to decide:

- Which messaging connection aliases for which you want to use enhanced logging. Enhanced logging is configured on a per messaging connection alias basis. You can configure enhanced logging for Universal Messaging connection alias for webMethods messaging or JMS connection alias.

For information about configuring enhanced logging for a Universal Messaging connection alias, see [“Creating a Universal Messaging Connection Alias” on page 238](#).

For information about configuring enhanced logging for a JMS connection alias, see [“Creating a JMS Connection Alias” on page 285](#).

- Whether Integration Server writes enhanced logging entries to the server.log or the messaging log.
- Whether to generate enhanced logging for message consumers and/or producers that use the messaging connection alias. You can further limit the message producers for which Integration Server performs enhanced logging by identify the specific destinations for which you want additional logging. Integration Server writes enhanced logging only for messages sent to the specific destinations using the messaging connection alias. Similarly, you can also instruct Integration Server to generate additional logging for all triggers that use the messaging connection alias to receive messages or just specific triggers.
- The logging level for enhanced logging. Where you set the logging level depends on whether Integration Server writes enhanced logging entries to the messaging log or the server log.
 - For the messaging log, you configure the logging level on the messaging audit logger. For information about configuring the messaging audit logger, see the *webMethods Audit Logging Guide*.
 - For the server log, you configure the logging level for the 0168 Messaging (Enhanced Logging) server log facility. You must set the facility to at least the Info logging level for Integration Server to write enhanced logging entries. For information about configuring the server.log see [“Specifying Amount and Type of Information to Include in the Server Log” on page 215](#).

Enhanced logging has a performance cost. The additional logging by Integration Server consumes resources and impacts performance negatively. The additional log entries can very quickly fill up disk space. Keep this information in mind when using enhanced logging for messaging.

Viewing the Messaging Log

You can use a text editor or Integration Server Administrator to view the messaging log. Note that Integration Server Administrator displays the current messaging log only. Integration Server Administrator does not include log entries from any archived messaging logs.

Integration Server Administrator also provides the ability to search the current messaging log for all entries that correspond to a particular UUID.

For more information about viewing the messaging log in Integration Server Administrator, see the *webMethods Audit Logging Guide* .

15 Setting Up Universal Messaging Client Logging

■ Universal Messaging Client Logging	342
■ Configuring Universal Messaging Client Logging	342
■ View the Universal Messaging Client Log	343

Universal Messaging Client Logging

Universal Messaging logs the activity of its client and its interaction with the Universal Messaging server. In particular, Universal Messaging logs responses from its server that deviate from what the client is expecting to happen. Typically these responses are in the form of exceptions. For example, if the client is not authorized to connect to the server, the server returns a security exception, and Universal Messaging logs that exception.

Integration Server offers the ability to configure this logging in Integration Server Administrator instead of in Universal Messaging.

Configuring Universal Messaging Client Logging

The Universal Messaging client log file is called `umClient.log`. You can configure the UM client logger to control the amount of detail in the Universal Messaging client log file .

> To configure Universal Messaging client logging

1. In Integration Server Administrator, go to the **Settings > Logging** page.
2. In the **Logger List**, click **UM client logger**.
3. Click **Edit UM Client Logger**.
4. Supply the following information

Field	Description
Log Level	Determines the information that is written to the log. Each level outputs log entries with that level or higher.
Log Size	Maximum size of the log file measured in megabytes (MB). When the log reaches this size , Integration Server rolls the file over to a backup called <code>umClient.log.number</code> and creates a new file.
Log Depth	Number of backup log files to keep on disk when using log rolling. When this number is reached, Integration Server deletes the oldest log file.

5. Click **Save Changes**.
6. If you changed the log size or depth, restart Integration Server.

Note:
This logger cannot be disabled.

View the Universal Messaging Client Log

To view the Universal Messaging client log, go to the *Integration Server_directory/instances/instanceName/logs* directory and open the *umClient.log* file in a text editor.

16 Configuring Endpoint Aliases for Web Services

■ About Web Service Endpoint Aliases	346
■ Creating an Endpoint Alias for a Provider Web Service Descriptor for Use with HTTP/S .	347
■ Creating an Endpoint Alias for a Consumer Web Service Descriptor for Use with HTTP/ S	355
■ Creating an Endpoint Alias for Message Addressing for Use with HTTP/S	367
■ Creating an Endpoint Alias for a Provider Web Service Descriptor for Use with JMS ...	373
■ Creating an Endpoint Alias for a Consumer Web Service Descriptor for Use with JMS .	379
■ Creating an Endpoint Alias for Message Addressing for Use with JMS	386
■ Timestamps in the WS-Security Header	392

About Web Service Endpoint Aliases

A web service endpoint alias represents the network address and, optionally, any security credentials to be used with web services. You can use the network address properties to enable dynamic addressing for web services. The security credentials can be used to control both transport-level and message-level security for web services.

In web service descriptors, an endpoint alias is associated with a binder. Integration Server uses a binder to collect the definitions for addresses, communication protocols, and data formats for a particular port type in one container. For more information about associating an endpoint alias with a binder, see *webMethods Service Development Help*.

For a consumer web service descriptor and its associated web service connectors (WSC), the alias information (including the addressing information and any security credentials), is used at run time to generate a request and invoke an operation of the web service.

For provider web service descriptors, the endpoint alias is used to construct the "location=" attribute of the address element (which is contained within the port element) when a WSDL file is requested from the web service. The security credentials might be used when constructing a response to a web service request.

When you create a provider web service descriptor, you can specify an existing endpoint alias, which will be displayed (and can be changed) from the default binder of the web service descriptors.

Integration Server uses message addressing endpoint aliases to send responses to endpoints other than the one that initiated or sent the request. That is, when WS-Addressing is enabled and the request SOAP message contains a non-anonymous ReplyTo or FaultTo endpoint, Integration Server uses the message addressing endpoint alias to determine the authentication details to be used to send a response to the ReplyTo and FaultTo endpoints.

An endpoint alias is usually created for one or more of the following reasons:

- **Dynamic endpoint addressing.** Because the actual value of the endpoint is looked up at run time, using an endpoint alias saves you from having to specify or change the server information each time you use the web service.
- **Security.** An endpoint alias is required in order to configure WS-Security for web service providers and consumers. For information about implementing WS-Security, see the information about WS-Security in *Web Services Developer's Guide*.
- **WS-Addressing.** Using the message addressing properties, you can specify the authentication credentials required to send a response to a different address than the request. The message addressing properties define the WS-Addressing information that can be attached to the SOAP message. For more information about WS-Addressing, see *Web Services Developer's Guide*.
- **WS-ReliableMessaging.** Reliable messaging properties ensure the reliable delivery of the message between the two endpoints (web service and client or reliable messaging source and destination). You can configure reliable messaging properties specific to a web service endpoint or at a global level for all web service endpoints defined in the Integration Server.

When you create web service endpoint aliases, keep the following points in mind:

- Alias names must be unique within the specified usage (provider or consumer) and protocol. This can result in multiple endpoint aliases with the same name. For example, you can have a provider alias named "aliasOne" for the HTTP protocol. You could also have a consumer alias named "aliasOne" for the HTTP protocol and a provider alias named "aliasOne" for the HTTPS protocol.
- Integration Server saves web service endpoint aliases at the following location:
Integration Server_directory \instances\instance_name\config\endpoints
- The host name and port are required for a provider HTTP/S web service endpoint alias, but are optional for a consumer HTTP/S web service endpoint alias.
- If the Integration Server on which a consumer web service descriptors resides sits behind a firewall and the web service request needs to be routed through a proxy server, you can assign a proxy alias to the consumer web service endpoint alias.
- You can identify default provider web service endpoint aliases for HTTP and HTTPS. If a provider web service descriptor contains a binder set to the default alias, Integration Server uses the information in the default alias when constructing the WSDL for the descriptor.

Creating an Endpoint Alias for a Provider Web Service Descriptor for Use with HTTP/S

When creating a web service endpoint alias for provider web service descriptor that uses an HTTP/S binder, you need to supply information that falls into the following categories:

- **Web Service Endpoint Alias.** Endpoint name, description, and transport type.
- **HTTP/S Transport Properties.** Server on which the web service resides.
- **WS Security Properties.** Information the SOAP processor needs to decrypt and verify the inbound SOAP request and/or encrypt and sign the outbound SOAP response and the details for adding the timestamp information.

Note:

WS-Security credentials such as private keys and public keys do not always need to be provided in a web service endpoint alias. If this information is not provided in the alias, Integration Server can obtain the information from other locations. For more information about usage and resolution order of certificates and keys for WS-Security, see the *Web Services Developer's Guide*.

- **Message Addressing Properties.** WS-Addressing information that Integration Server uses to generate the WS-Addressing headers of the SOAP requests and responses. This includes the destination address of a message or fault and the authentication credentials required to send a response to a different address than the one from which request was received.
- **Reliable Messaging Properties.** Reliable messaging information specific to the web service endpoint. By default, Integration Server applies the reliable messaging configuration defined on the **Settings > Web Services > Reliable Messaging > Edit Configuration** page to all web service providers and consumers. If you want to override the server-level reliable messaging

configuration for a specific web service provider or consumer, define reliable messaging properties for the associated web service endpoint alias.

➤ **To create a WS provider web service endpoint alias for use with HTTP/S**

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Create Web Service Endpoint Alias**.
4. Under **Web Service Endpoint Alias Properties**, provide the following information:

In this field	Specify
Alias	A name for the provider web service endpoint alias. The alias name cannot include the following illegal characters: <code># © \ & @ ^ ! % * : \$. / \ \ ` ; , ~ + =) (} {] [> < "</code>
Description	A description for the endpoint alias.
Type	Provider
Transport Type	Specify the transport protocol used to access the web service. Select one of the following: <ul style="list-style-type: none"> ■ HTTP ■ HTTPS

5. Under **TransportType Transport Properties**, provide the following information:

In this field	Specify
Host Name or IP Address	Host name or IP address of the Integration Server for which you are creating an alias. If the host Integration Server is fronted by a proxy, specify the host name or IP address of the proxy server.
Port	An active HTTP or HTTPS listener port defined on the Integration Server specified in the Host Name or IP Address field. If the host Integration Server is fronted by a proxy, specify the port for the proxy server.

6. Under **WS Security Properties**, if the inbound SOAP request must be decrypted and/or the outbound SOAP request must be signed, do the following:

In this field	Specify
Keystore Alias	Alias of the keystore containing the private key used to decrypt the inbound SOAP request or sign the outbound SOAP response.
	Important: The provider must have already given the consumer the corresponding public key.
Key Alias	Alias of the private key used to decrypt the request or sign the response. The key must be in the keystore specified in Keystore Alias .

7. Under **WS Security Properties**, if the signing certificate chain of an inbound signed SOAP message has to be *validated*, specify the following:

In this field	Specify
Truststore Alias	The alias for the truststore that contains the list of CA certificates that Integration Server uses to validate the trust relationship.

8. Under **WS Security Properties**, set the timestamp properties that Integration Server uses when working with timestamps.

In this field	Specify
Timestamp Precision	Whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss:SSS'Z'</code> . If you set the precision to seconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss'Z'</code> . If you do not select a precision value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.
Timestamp Time to Live	The time-to-live value for the outbound message in seconds. Integration Server uses the Timestamp Time to Live value to set the expiry time in the Timestamp element of outbound messages. The time-to-live value must be an integer greater than 0. If you do not specify a Timestamp Time to Live value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.
Timestamp Maximum Skew	The maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed. Specify a positive integer or zero. Integration Server uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. Integration Server validates

In this field	Specify
	<p>the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>
Username Token TTL	This is the permitted time difference, in seconds, between the time when the UsernameToken was created (as provided in the <code>wsu:Created</code> element) and the time when it reaches the server. Requests that exceed this limit are rejected by the server. The default value is 300.
Username Token Future TTL	It is possible that the <code>wsu:Created</code> element has a timestamp that is in the future. The server considers such requests as valid If the time at which the request was created does not exceed the time at which it reaches the server by the value (in seconds) given in this setting. The default value is 60.

Note:

The **Username Token TTL** and **Username Token Future TTL** configurations can also be set at the global level using the `watt.server.ws.security.usernameTokenTTL` and the `watt.server.ws.security.usernameTokenFutureTTL` server configuration properties. However, if there is a configuration setting at the web services endpoint level, the server will ignore the global property. For more information about the global properties, see [“watt.server.” on page 1053](#).

For more information about timestamps in the WS-Security header, see [“Timestamps in the WS-Security Header” on page 392](#).

- Under **Kerberos Properties**, provide the following Kerberos-related details that will be used for all providers that use this endpoint alias.

Note:

These fields are available only for provider endpoint aliases using the HTTPS transport type.

In this field	Specify
JAAS Context	<p>The custom JAAS context used for Kerberos authentication.</p> <p>In the following example, JAAS Context is <code>WS_KERBEROS_INBOUND</code>:</p> <pre>WS_KERBEROS_INBOUND { com.sun.security.auth.module.Krb5LoginModule required refreshKrb5Config=true storeKey=true isInitiator=false debug=true;</pre>

In this field	Specify						
	<code>};</code>						
	The <code>is_jaas.cnf</code> file distributed with Integration Server includes a JAAS context named <code>IS_KERBEROS_INBOUND</code> that can be used with inbound requests.						
Principal	The name of the principal to use for Kerberos authentication.						
Principal Password	The password for the principal that is used to authenticate the principal to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for authorization. The passwords may be encrypted using different encryption algorithms. If the JAAS login context contains <code>useKeyTab=false</code> , you must specify the principal password.						
Retype Principal Password	The above principal password.						
Service Principal Name Format	Select the format in which you want to specify the principal name of the service that is registered with the principal database.						
	<table> <thead> <tr> <th>Select</th> <th>To</th> </tr> </thead> <tbody> <tr> <td>host-based</td> <td>Represent the principal name using the service name and the hostname, where hostname is the host computer. This is the default.</td> </tr> <tr> <td>username</td> <td>Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.</td> </tr> </tbody> </table>	Select	To	host-based	Represent the principal name using the service name and the hostname, where hostname is the host computer. This is the default.	username	Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.
Select	To						
host-based	Represent the principal name using the service name and the hostname, where hostname is the host computer. This is the default.						
username	Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.						
Service Principal Name	The name of the principal for the service that the Kerberos client wants to access. This can be obtained from the WSDL document published by the provider of the Kerberos service. Specify the Service Principal Name in the following format: <code>principal-name.instance-name@realm-name</code>						

10. Under **Message Addressing Properties**, provide the following addressing information relating to the delivery of the message. The message addressing properties define the addressing information that can be attached to the SOAP message.

In this field	Specify
To	URI of the destination of the SOAP message.

In this field **Specify**

In the **Reference Parameters** field, specify additional parameters, if any, that correspond to `<wsa:ReferenceParameters>` properties of the endpoint reference to which the message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the **Metadata Elements** field. You can also specify **Extensible Elements**, which are elements other than those specified as part of the **Metadata** and **Reference Parameters**.

You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.

Response Map Address to which the provider will send the reply or fault message and the corresponding message addressing alias. Integration Server retrieves the authentication details needed to send the response from the message addressing alias mapped to the address.

In the **Address** field, specify the URI to which the provider will send the reply or the fault message.

From the **Message Addressing Alias** list, select the message addressing endpoint alias from which Integration Server will retrieve the authentication details. Integration Server uses the authentication details to send the response to the ReplyTo or FaultTo endpoints.

Click the '+' icon to add more rows and the 'x' icon to delete the rows.

11. Under **Reliable Messaging Properties**, check **Enable** to provide reliable messaging information specific to the endpoint alias you are creating.
12. Provide the following reliable-messaging information to ensure reliable delivery of the message between a reliable messaging source and destination.

In this field **Specify**

Retransmission Interval The time interval (in milliseconds) for which a reliable messaging source waits for an acknowledgment from the reliable messaging destination before retransmitting the SOAP message. The default is 6000 milliseconds.

Acknowledgement Interval The time interval (in milliseconds) for which the reliable messaging destination waits before sending an acknowledgment for a message sequence. Messages of the same sequence received within the specified acknowledgment interval are acknowledged in one batch. If there are no other messages to be sent to the acknowledgment endpoint within the time specified as the acknowledgment interval, the acknowledgment is sent as a stand-alone message.

The default is 3000 milliseconds.

In this field	Specify				
Exponential Backoff	Whether to use the exponential backoff algorithm to adjust the retransmission interval of unacknowledged messages. Adjusting the time interval between retransmission attempts ensures that a reliable messaging destination does not get flooded with a large number of retransmitted messages.				
	<table border="0" style="width: 100%;"> <tr> <td style="text-align: left; width: 15%;">Select</td> <td style="text-align: left;">To</td> </tr> </table>	Select	To		
Select	To				
	<table border="0" style="width: 100%;"> <tr> <td style="vertical-align: top; width: 15%;">true</td> <td>Increase the successive retransmission intervals exponentially, based on the specified retransmission interval. For example, if the specified retransmission interval is 2 seconds, and the exponential backoff value is set to true, successive retransmission intervals will be 2, 4, 8, 16, 32, and so on if messages continue to be unacknowledged. This is the default.</td> </tr> <tr> <td style="vertical-align: top;">false</td> <td>Use the same time interval specified in the Retransmission Interval field for all retransmissions.</td> </tr> </table>	true	Increase the successive retransmission intervals exponentially, based on the specified retransmission interval. For example, if the specified retransmission interval is 2 seconds, and the exponential backoff value is set to true, successive retransmission intervals will be 2, 4, 8, 16, 32, and so on if messages continue to be unacknowledged. This is the default.	false	Use the same time interval specified in the Retransmission Interval field for all retransmissions.
true	Increase the successive retransmission intervals exponentially, based on the specified retransmission interval. For example, if the specified retransmission interval is 2 seconds, and the exponential backoff value is set to true, successive retransmission intervals will be 2, 4, 8, 16, 32, and so on if messages continue to be unacknowledged. This is the default.				
false	Use the same time interval specified in the Retransmission Interval field for all retransmissions.				
Maximum Retransmission Count	The number of times the reliable messaging source must retransmit a message if an acknowledgement is not received from the reliable messaging destination. To specify that there is no limit to the number of retransmission attempts, set the value of Maximum Retransmission Count to -1. The default is 10.				

13. Click **Save Changes**.

Setting a Default Endpoint Alias for Provider Web Service Descriptors

For the HTTP and HTTPS protocols, you can identify a provider web service endpoint alias as the default provider endpoint alias for each protocol. When the default provider endpoint alias is assigned to a binder in a provider web service descriptor, Integration Server uses the information in the alias when constructing the WSDL for the descriptor and during run-time processing. Simply changing the default provider endpoint alias for a protocol changes the information used to generate the WSDL and the information used for run-time processing. You do not need to edit the binder in the web service descriptor to specify a different alias.

Integration Server uses the default provider endpoint alias in the following situations:

- When constructing the WSDL for the a provider web service descriptor that contains a binder with a **Port alias** property set to DEFAULT(*aliasName*) or binder that does not explicitly set an alias for the **Port alias** property.

- During run-time processing for provider web service descriptor that contains a binder with a **Port alias** property set to DEFAULT(*aliasName*) or a binder that does not explicitly set an alias for the **Port alias** property.
- As an available alias when creating the endpoint for a service first provider web service
- As an available alias when setting the endpoint for a binder.
- When creating the binders for a WSDL first provider web service descriptor generated from a WSDL document with an HTTP or HTTPS binding. Integration Server assigns the default provider endpoint alias of the transport protocol to the binder. Integration Server uses the information from the alias during WSDL generation and run-time processing.

Note:

If the binder in a provider web service descriptor does not specify a value for the **Port alias** property and there is not a default provider endpoint alias for the protocol used by the binder, when Integration Server generates the WSDL document for the web service descriptor, Integration Server sets the "location=" attribute of the soap:address element to localhost:*primaryPort*.

Keep the following points in mind when setting a default provider endpoint alias for use with provider web service descriptors:

- You can set a default provider endpoint alias for provider web services only.
- You can set a default provider endpoint alias for the HTTP and HTTPS protocols. You cannot set a default endpoint alias for JMS.
- Integration Server does not require that a default provider endpoint alias be set. If there is no default alias for a protocol, the **Port alias** property for a binder in a provider web service descriptor lists a blank row as a possible value. If you select the blank row and later specify a default alias for the protocol used by the binder, Integration Server uses the information in the default provider endpoint alias when generating the WSDL document and during run-time processing for the web service descriptor. That is, once a default provider endpoint alias is set for a protocol, any previously blank **Port alias** properties are effectively set to DEFAULT(*aliasName*) for binders that use that protocol.
- You cannot delete a web service endpoint alias used as a default alias.

➤ To set a default endpoint alias for provider web service descriptors

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Set Default Provider Endpoint Alias**.
4. Under **Select Default Provider Endpoint Aliases**, do one or more of the following:
 - In the **HTTP** list, select the alias to use as the default endpoint alias for the HTTP protocol. If you do not want to set a default endpoint alias for HTTP, select the blank row.

- In the **HTTPS** list, select the alias to use as the default endpoint alias for the HTTPS protocol. If you do not want to set a default endpoint alias for HTTPS, select the blank row.

5. Click **Update**.

Integration Server updates the WSDL document for any provider web service descriptor with a binder that uses the default provider endpoint alias.

Note:

After changing the default provider endpoint alias, consumers should refresh any existing web service clients generated from a WSDL document for a web service descriptor that uses the default provider endpoint alias. Refreshing the web service client using the updated WSDL document enables the client to make use of the changed endpoint information.

Creating an Endpoint Alias for a Consumer Web Service Descriptor for Use with HTTP/S

When you create an HTTP/S web service endpoint alias for use with consumer web service descriptors, you need to supply information that falls into the following categories:

- **Web Service Endpoint Alias.** Endpoint name, description, and transport type.
- **HTTP/S Transport Properties.** Optional. The host and port used to build the endpoint URL. If the web service provider requires transport-based authentication, these properties specify the authentication credentials to be added to the HTTP/S header. For HTTPS transport, these properties specify the keystore alias and key alias of the private key used for SSL communication with the web service provider. If the web service request must be routed through a proxy server, these properties specify the proxy server alias for the proxy server through which Integration Server routes the HTTP/S request.
- **WS Security Properties.** Information for the WS-Security header as determined by the security policy for the web service. A web service security policy can require that:
 - SOAP message requests include a Username token.
 - SOAP message responses be decrypted.
 - SOAP message requests be signed.
 - X.509 authentication be supported.
 - A Timestamp element be added to the security header.

Note:

WS-Security credentials such as private keys and public keys do not always need to be provided in a web service endpoint alias. If this information is not provided in the alias, Integration Server can obtain the information from other locations. For more information about usage and resolution order of certificates and keys for WS-Security, see the *Web Services Developer's Guide*.

- **Message Addressing Properties.** Addressing information about the response delivery. This information includes the reply endpoint where the replies should be sent, the fault endpoint that specifies where the faults should be sent, and optional metadata (such as WSDL or WS-Policy) about the service. This also includes additional parameters, called Reference Parameters, that Integration Server uses to route the message to the destination.
- **Reliable Messaging Properties.** Provides reliable messaging information specific to the web service endpoint. By default, Integration Server applies the reliable messaging configuration defined on the **Settings > Web Services > Reliable Messaging > Edit Configuration** page to all web service providers and consumers. If you want to override the server-level reliable messaging configuration for a specific web service provider or consumer, define reliable messaging properties for the associated web service endpoint alias.

➤ **To create a consumer web service endpoint alias for use with HTTP/S**

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Create Web Service Endpoint Alias**.
4. Under **Web Service Endpoint Alias Properties**, provide the following information:

In this field	Specify
Alias	A name for the provider web service endpoint alias. The alias name cannot include the following illegal characters: # ©\ & @ ^ ! % * : \$. / \ \ ` ; , ~ + =) (} {] [> < "
Description	A description for the endpoint alias.
Type	Consumer
Transport Type	Specify the transport protocol used to access the web service. Select one of the following: <ul style="list-style-type: none"> ■ HTTP ■ HTTPS
Execute ACL	ACL that governs which user groups on your server can use this web service endpoint alias. Select an ACL from the drop down list. By default, only members of groups governed by the Internal ACL can use this alias. Integration Server performs the ACL check only if the specific endpoint alias is used as the value of the <i>endpointAlias</i> input parameter in a web service connector or the <i>pub.client:soapClient</i> service. Integration Server does

In this field	Specify
	not perform the ACL check when the consumer web service endpoint alias is assigned to a binder used by the web service connector.

5. Under *TransportType* **Transport Properties**, provide the following information if you want to overwrite the host and/or port information in the WSDL with the host and/or port information in the web service endpoint alias. For more information about how Integration Server builds the endpoint URL, see *webMethods Service Development Help*.

In this field	Specify										
Host Name or IP Address	Host name or IP address of the server on which the web service resides.										
Port Number	An active HTTP or HTTPS type listener port defined on the host server specified in the Host Name or IP Address field.										
Authentication Type	Specify the type of authentication you want to use to authenticate the consumer.										
	<table border="1"> <thead> <tr> <th>Select...</th> <th>To...</th> </tr> </thead> <tbody> <tr> <td>Basic</td> <td>Use basic authentication (user name and password) to authenticate the consumer.</td> </tr> <tr> <td>Digest</td> <td>Use password digest to authenticate the consumer.</td> </tr> <tr> <td>NTLM</td> <td>Use NTLM authentication so that clients that are already logged into a domain can be authenticated using their existing credentials.</td> </tr> <tr> <td>Kerberos</td> <td>Use Kerberos authentication for the web service at the transport level. When you select this option an additional section Kerberos Transport Properties is added to this page.</td> </tr> </tbody> </table>	Select...	To...	Basic	Use basic authentication (user name and password) to authenticate the consumer.	Digest	Use password digest to authenticate the consumer.	NTLM	Use NTLM authentication so that clients that are already logged into a domain can be authenticated using their existing credentials.	Kerberos	Use Kerberos authentication for the web service at the transport level. When you select this option an additional section Kerberos Transport Properties is added to this page.
Select...	To...										
Basic	Use basic authentication (user name and password) to authenticate the consumer.										
Digest	Use password digest to authenticate the consumer.										
NTLM	Use NTLM authentication so that clients that are already logged into a domain can be authenticated using their existing credentials.										
Kerberos	Use Kerberos authentication for the web service at the transport level. When you select this option an additional section Kerberos Transport Properties is added to this page.										

6. If you are configuring the web service endpoint for transport-based authentication such as HTTPS, specify all or a combination of the following optional fields:

In this field	Specify
User Name	User name used to authenticate the consumer at the HTTP or HTTPS transport level on the host server.
Password	The password used to authenticate the consumer on the host server.
Retype Password	Re-enter the above password.

In this field	Specify
Keystore Alias	Alias to the keystore that contains the private key used to connect to the web service host securely. This field applies to the HTTPS transport type only.
Key Alias	Alias to the key in the keystore that contains the private key used to connect to the web service host securely. The key must be in the keystore specified in Keystore Alias . This field applies to the HTTPS transport type only.

7. If web service requests must be sent through a proxy server, in the **Proxy Alias** field, do one of the following to specify which proxy server Integration Server uses:
- If you want Integration Server to use a particular proxy server, select the alias for that proxy server. Integration Server lists all the configured HTTP/S and SOCKS proxy aliases in the **Proxy Alias** field.
 - If you want Integration Server to use the default proxy server, leave this field blank.

For more information about how Integration Server uses proxy servers when sending requests, see [“How Integration Server Uses Proxy Servers” on page 119](#).

8. The **Kerberos Transport Properties** section enables you to configure Kerberos authentication at the transport-level. You can provide Kerberos-related details that will be used for all web service requests by providers that use this endpoint alias.

Note:

Kerberos authentication is available at the transport level or message level for HTTPS, but only at the transport level for HTTP.

The following fields are available for consumer endpoint aliases when you select **Kerberos** as the HTTP transport **Authentication Type**.

Field	Description
JAAS Context	Specify the custom JAAS context used for Kerberos authentication. In the following example, JAAS Context is <code>KerberosClient</code> : <pre>KerberosClient { com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true keyTab=alice.keytab; };</pre>
Principal	Specify the name of the principal to use for Kerberos authentication.

Field	Description						
Principal Password	Specify the password for the principal that is used to authenticate the principal to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for authorization. The passwords may be encrypted using different encryption algorithms. If the JAAS login context contains <code>useKeyTab=false</code> , you must specify the principal password.						
Retype Principal Password	Re-enter the above principal password.						
Service Principal Name Format	Select the format in which you want to specify the principal name of the service that is registered with the principal database.						
	<table border="1"> <thead> <tr> <th>Select...</th> <th>To...</th> </tr> </thead> <tbody> <tr> <td>host-based</td> <td>Represent the principal name using the service name and the host name, where host name is the host computer. Note: Currently, this option is disabled. Integration Server supports only username.</td> </tr> <tr> <td>username</td> <td>Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.</td> </tr> </tbody> </table>	Select...	To...	host-based	Represent the principal name using the service name and the host name, where host name is the host computer. Note: Currently, this option is disabled. Integration Server supports only username.	username	Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.
Select...	To...						
host-based	Represent the principal name using the service name and the host name, where host name is the host computer. Note: Currently, this option is disabled. Integration Server supports only username.						
username	Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.						
Service Principal Name	Specify the service that the Kerberos client wants to access. This can be obtained from the WSDL document published by the provider of Kerberos service. Specify the Service Principal Name in the following format: <i>principal-name.instance-name@realm-name</i>						

9. Under **WS Security Properties**, provide the following information if the WS-Security policy for this consumer web service descriptor requires that SOAP message requests include a UsernameToken, enter values for the following fields:

In this field	Specify
User Name	The user name to include with the UsernameToken.
Password	The password to include with the UsernameToken (must be plain text).
Retype Password	Re-enter the above password.

10. If the security policy (or policies) that will be used by this web service requires its requests to be *signed*, requires an X.509 authentication token to be included, or requires that SOAP message responses be encrypted, specify the following:

In this field	Specify
Keystore Alias	Alias to the keystore that contains the private key used to: <ul style="list-style-type: none">■ Sign outbound SOAP requests■ Include an X.509 authentication token for outbound SOAP requests■ Decrypt inbound SOAP responses <div style="background-color: #f0f0f0; padding: 5px;">Important: To verify messages from this consumer, the web services provider must have a copy of the corresponding public key.</div>
Key Alias	Alias to the private key used to sign and/or include X.509 authentication token for outbound SOAP messages and/or decrypt inbound SOAP responses. The key must be in the keystore specified in Keystore Alias .

11. Under **WS Security Properties**, specify the provider's certificate file. This certificate is used to encrypt the outbound SOAP request and/or verify the inbound SOAP response.

In this field	Specify
Partner's Certificate	The path and file name of the provider's certificate, which contains its public key.

12. Under **WS Security Properties**, if the security policy (or policies) that will be used by this web services consumer requires that responses be *validated* by a trusted authority, specify the following:

In this field	Specify
Partner's Certificate	Path and file name of the file containing the provider's certificate.
Truststore Alias	The alias for the truststore that contains the list of CA certificates that Integration Server uses to validate the trust relationship.

13. Under **WS Security Properties**, configure how Integration Server handles timestamps in the security headers.

In this field	Specify
Timestamp Precision	Whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, Integration Server uses the timestamp

In this field	Specify
	<p>format <code>yyyy-MM-dd'T'HH:mm:ss:SSS'Z'</code>. If you set the precision to seconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss'Z'</code>.</p> <p>If you do not select a precision value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.</p>
Timestamp Time to Live	<p>The time-to-live value for the outbound message in seconds. Integration Server uses the Timestamp Time to Live value to set the expiry time in the <code>Timestamp</code> element of outbound messages. The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a time-to-live value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p>
Timestamp Maximum Skew	<p>The maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed. Specify a positive integer or zero.</p> <p>Integration Server uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. Integration Server validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>
Username Token TTL	<p>This is the permitted time difference, in seconds, between the time when the <code>UsernameToken</code> was created (as provided in the <code>wsu:Created</code> element) and the time when it reaches the server. Requests that exceed this limit are rejected by the server. The default value is 300.</p>
Username Token Future TTL	<p>It is possible that the <code>wsu:Created</code> element has a timestamp that is in the future. The server considers such requests as valid If the time at which the request was created does not exceed the time at which it reaches the server by the value (in seconds) given in this setting. The default value is 60.</p>

Note:

The **Username Token TTL** and **Username Token Future TTL** configurations can also be set at the global level using the `watt.server.ws.security.usernameTokenTTL` and the `watt.server.ws.security.usernameTokenFutureTTL` server configuration properties. However, if there is a configuration setting at the web services endpoint level, the server will ignore the global property. For more information about the global properties, see [“watt.server.” on page 1053](#).

For more information about timestamps in the WS-Security header, see [“Timestamps in the WS-Security Header” on page 392](#).

14. Under **Kerberos Properties**, provide the following Kerberos-related details that will be used for all web service requests that use this endpoint alias.

Note:

These fields are available only for consumer endpoint aliases using HTTPS transport type.

In this field	Specify
JAAS Context	<p>The custom JAAS context used for Kerberos authentication.</p> <p>In the following example, JAAS Context is <code>KerberosClient</code>:</p> <pre style="background-color: #f0f0f0; padding: 5px;">KerberosClient { com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true keyTab=alice.keytab; };</pre> <p>The <code>is_jaas.cnf</code> file distributed with Integration Server includes a JAAS context named <code>IS_KERBEROS_OUTBOUND</code> that can be used with inbound requests.</p>
Principal	The name of the principal to use for Kerberos authentication.
Principal Password	The password for the principal that is used to authenticate the principal to the KDC. Specify the principal password if you do not want to use the keytab file that contains the principals and their passwords for authorization. The passwords may be encrypted using different encryption algorithms. If the JAAS login context contains <code>useKeyTab=false</code> , you must specify the principal password.
Retype Principal Password	The above principal password.
Service Principal Name Format	Select the format in which you want to specify the principal name of the service that is registered with the principal database.
 Select	To
 host-based	<p>Represent the principal name using the service name and the hostname, where hostname is the host computer.</p> <p>This is the default.</p>
 username	Represent the principal name as a named user defined in the LDAP or central user directory used for authentication to the KDC.

In this field **Specify**

Service Principal Name The name of the principal for the service that the Kerberos client wants to access. This can be obtained from the WSDL document published by the provider of the Kerberos service. Specify the Service Principal Name in the following format:

```
principal-name.instance-name@realm-name
```

15. Under **Message Addressing Properties**, provide the following addressing information relating to the delivery of a message to a web service. The message addressing properties define the addressing information that can be attached to the SOAP message.

In this field **Specify**

Must Understand Whether the recipients (the actor or role to which the header is targeted) are required to process the WS-Addressing headers. Recipients that cannot process a mandatory WS-Addressing header reject the message and return a SOAP fault.

Must Understand determines the `mustUnderstand` attribute of the WS-Addressing headers.

Select	To
True	Indicate that processing the WS-Addressing headers is required by the recipients. If you select True for Must Understand and the SOAP node receives a header that it does not understand or cannot process, it returns a fault.
False	Indicate that processing the WS-Addressing headers is optional. This is the default.

Note:

In SOAP 1.1, the values of the `mustUnderstand` attribute were 0 and 1 instead of True and False; however, Integration Server processes both sets of values the same way and performs any necessary conversions.

For more information about the `mustUnderstand` and actor attributes in SOAP 1.1, see the *Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000* specification.

For more information about the `mustUnderstand` and role attributes in SOAP 1.2, see the *Simple Object Access Protocol (SOAP) 1.2 specification*.

In this field	Specify										
Role	<p>Target of the WS-Addressing headers in the SOAP message. Role determines the value of the role attribute for the WS-Addressing headers. The actor or role attribute specifies a URI for the recipient of WS-Addressing header entries.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: In SOAP 1.1, the role attribute is named actor; however, Integration Server processes both names the same and performs any necessary conversions.</p> </div> <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left; border-bottom: 1px solid black;">Select</th> <th style="text-align: left; border-bottom: 1px solid black;">To</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">Ultimate Receiver</td> <td>Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.</td> </tr> <tr> <td style="vertical-align: top;">Next</td> <td> Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next" ■ For SOAP 1.1: "http://schemas.xmlsoap.org/soap/actor/next" </td> </tr> <tr> <td style="vertical-align: top;">None</td> <td> Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/none" ■ For SOAP 1.1: "http://www.w3.org/2003/05/soap-envelope/role/none" </td> </tr> <tr> <td style="vertical-align: top;">Other</td> <td>Specify the target of the header. Typically, this will be a URI.</td> </tr> </tbody> </table>	Select	To	Ultimate Receiver	Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.	Next	Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next" ■ For SOAP 1.1: "http://schemas.xmlsoap.org/soap/actor/next" 	None	Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/none" ■ For SOAP 1.1: "http://www.w3.org/2003/05/soap-envelope/role/none" 	Other	Specify the target of the header. Typically, this will be a URI.
Select	To										
Ultimate Receiver	Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.										
Next	Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next" ■ For SOAP 1.1: "http://schemas.xmlsoap.org/soap/actor/next" 										
None	Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/none" ■ For SOAP 1.1: "http://www.w3.org/2003/05/soap-envelope/role/none" 										
Other	Specify the target of the header. Typically, this will be a URI.										
To	<p>URI of the destination of the SOAP message.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the request is addressed. You can specify more than one reference parameter. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>										
From	<p>URI of the source of the SOAP message. Enter the URI in the Address field.</p> <p>Optionally, in the Reference Parameters field, specify any additional parameters that are necessary to route the message to the destination. You can also specify optional metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can specify more than one reference parameter and</p>										

In this field	<p>Specify</p> <p>metadata element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
ReplyTo	<p>URI of the destination to which the web service sends a response (reply) message. This property is optional.</p> <p>If this value is not specified, the default values for this URI depends on the WS-Addressing policy attached to the web service descriptor. For a consumer endpoint alias, it defaults to:</p> <ul style="list-style-type: none"> ■ http://www.w3.org/2005/08/addressing/anonymous for Final version of WS-Addressing. ■ http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous for Submission version of WS-Addressing. <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the response message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
FaultTo	<p>URI to which the SOAP fault messages are to be routed. This property is optional.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the fault message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>

16. Under **Reliable Messaging Properties**, check **Enable** to provide reliable messaging information specific to the endpoint alias you are creating.

17. Provide the following reliable-messaging information to ensure reliable delivery of the message between a reliable messaging source and destination.

In this field **Specify**

Retransmission Interval The time interval (in milliseconds) for which a reliable messaging source waits for an acknowledgement from the reliable messaging destination before retransmitting the SOAP message. The default is 6000 milliseconds.

Acknowledgement Interval The time interval (in milliseconds) for which the reliable messaging destination waits before sending an acknowledgement for a message sequence. Messages of the same sequence received within the specified acknowledgement interval are acknowledged in one batch. If there are no other messages to be sent to the acknowledgement endpoint within the time specified as the acknowledgement interval, the acknowledgement is sent as a stand-alone message.

The default is 3000 milliseconds.

Exponential Backoff Whether to use the exponential backoff algorithm to adjust the retransmission interval of unacknowledged messages. Adjusting the time interval between retransmission attempts ensures that a reliable messaging destination does not get flooded with a large number of retransmitted messages.

Select	To
true	Increase the successive retransmission intervals exponentially, based on the specified retransmission interval. For example, if the specified retransmission interval is 2 seconds, and the exponential backoff value is set to true, successive retransmission intervals will be 2, 4, 8, 16, 32, and so on if messages continue to be unacknowledged. This is the default.
false	Use the same time interval specified in the Retransmission Interval field for all retransmissions.

Maximum Retransmission Count The number of times the reliable messaging source must retransmit a message if an acknowledgement is not received from the reliable messaging destination. To specify that there is no limit to the number of retransmission attempts, set the value of **Maximum Retransmission Count** to -1. The default is 10.

18. Click **Save Changes**.

Creating an Endpoint Alias for Message Addressing for Use with HTTP/S

When creating an HTTP/S web service endpoint alias for message addressing, the information you need to supply falls into the following categories:

- **Web Service Endpoint Alias.** Identifies the endpoint name, description, and transport type.
- **HTTP/S Transport Properties.** Specifies the authentication details that Integration Server uses to send responses. For HTTPS transport, also specifies the keystore alias and key alias of the private key used for SSL communication with the receiver of the SOAP response.

If the web service response must be routed through a proxy server, specify the proxy server alias for the proxy server through which Integration Server routes the HTTP/S message.

- **WS Security Properties.** Provides information for the WS-Security header as determined by the security policy for the web service.

Note:

WS-Security credentials such as private keys and public keys do not always need to be provided in a web service endpoint alias. If this information is not provided in the alias, Integration Server can obtain the information from other locations. For more information about usage and resolution order of certificates and keys for WS-Security, see the *Web Services Developer's Guide*.

- **Message Addressing Properties.** Provides addressing information relating to the delivery of the response message. This includes the reply endpoint where the replies should be sent, the fault endpoint that specifies where the faults should be sent, and optional metadata (such as WSDL or WS-Policy) about the service. This also includes additional parameters, called Reference Parameters, that Integration Server uses to route the message to the destination.

Note:

You cannot delete a message addressing endpoint alias if a web service endpoint alias for provider web service descriptor is using the message addressing endpoint alias as a part of its response map.

➤ To create a message addressing web service endpoint alias for use with HTTP/S

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Create Web Service Endpoint Alias**.
4. Under **Web Service Endpoint Alias Properties**, provide the following information:

In this field	Specify
Alias	A name for the message addressing web service endpoint alias. The alias name cannot include the following illegal characters: # © \ & @ ^ ! % * : \$. / \ \ ` ; , ~ + =) (} { } [> < "
Description	A description for the endpoint alias.
Type	Message Addressing
Transport Type	Specify the transport protocol used to access the web service. Select one of the following: <ul style="list-style-type: none">■ HTTP■ HTTPS

5. If you are configuring the web service endpoint for transport-based authentication such as HTTPS, specify all or a combination of the following optional fields under *TransportType* **Transport Properties**:

In this field	Specify								
Authentication Type	Specify the type of authentication you want to use to authenticate the consumer. <table><thead><tr><th>Select</th><th>To</th></tr></thead><tbody><tr><td>Basic</td><td>Use basic authentication (user name and password) to authenticate the consumer.</td></tr><tr><td>Digest</td><td>Use password digest to authenticate the consumer.</td></tr><tr><td>NTLM</td><td>Use NTLM authentication so that clients that are already logged into a domain can be authenticated using their existing credentials.</td></tr></tbody></table>	Select	To	Basic	Use basic authentication (user name and password) to authenticate the consumer.	Digest	Use password digest to authenticate the consumer.	NTLM	Use NTLM authentication so that clients that are already logged into a domain can be authenticated using their existing credentials.
Select	To								
Basic	Use basic authentication (user name and password) to authenticate the consumer.								
Digest	Use password digest to authenticate the consumer.								
NTLM	Use NTLM authentication so that clients that are already logged into a domain can be authenticated using their existing credentials.								
User Name	User name used to authenticate the provider at the HTTP or HTTPS transport level on the host server.								
Password	The password used to authenticate the provider on the host server.								
Retype Password	Re-enter the above password.								
Keystore Alias	Alias to the keystore that contains the private key used to connect to the web service host securely. This field applies to the HTTPS transport type only.								

In this field	Specify
Key Alias	Alias to the key in the keystore that contains the private key used to connect to the web service host securely. The key must be in the keystore specified in Keystore Alias .

This field applies to the HTTPS transport type only.

- If web service responses must be sent through a proxy server, in the **Proxy Alias** field, do one of the following to specify which proxy server Integration Server uses:
 - If you want Integration Server to use a particular proxy server, specify the alias for that proxy server. Integration Server lists all the configured HTTP/S and SOCKS proxy aliases in the **Proxy Alias** field.
 - If you want Integration Server to use the default proxy server, leave this field blank.

For more information about how Integration Server uses proxy servers when sending responses, see [“How Integration Server Uses Proxy Servers” on page 119](#).

- Under **WS Security Properties**, specify the certificate file of the receiver of the SOAP response. This certificate is used to encrypt the outbound SOAP response and/or verify the inbound SOAP response.

In this field	Specify
Partner's Certificate	The path and file name of the certificate of the receiver of the SOAP response, which contains its public key.

- Under **WS Security Properties**, specify the following if the security policy (or policies) that will be used by this web service requires its responses to be *signed*, requires an X.509 authentication token to be included, or requires that SOAP message responses be encrypted:

In this field	Specify
Keystore Alias	Alias to the keystore that contains the private key used to: <ul style="list-style-type: none"> Sign outbound SOAP responses Include an X.509 authentication token for outbound SOAP responses
	<p>Important: To verify response messages from this web service, the receiver must have the corresponding public key.</p>
Key Alias	Alias to the private key used to sign and/or include X.509 authentication token for outbound SOAP messages. The key must be in the keystore specified in Keystore Alias .

9. Under **WS Security Properties**, configure how Integration Server handles timestamps in the security headers.

In this field	Specify
Timestamp Precision	<p>Whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss:SSS'Z'</code>. If you set the precision to seconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss'Z'</code>.</p> <p>If you do not select a precision value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.</p>
Timestamp Time to Live	<p>The time-to-live value for the outbound message in seconds. Integration Server uses the Timestamp Time to Live value to set the expiry time in the <code>Timestamp</code> element of outbound messages. The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a time-to-live value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p>
Timestamp Maximum Skew	<p>The maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed. Specify a positive integer or zero.</p> <p>Integration Server uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. Integration Server validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>

For more information about timestamps in the WS-Security header, see [“Timestamps in the WS-Security Header” on page 392](#).

10. Under **Message Addressing Properties**, provide the following addressing information relating to the delivery of a SOAP response to the receiver. The message addressing properties define the addressing information that can be attached to the SOAP message.

In this field	Specify
Must Understand	<p>Whether the recipients (the actor or role to which the header is targeted) are required to process the WS-Addressing headers. Recipients that cannot process a mandatory WS-Addressing header reject the message and return a SOAP fault.</p>

In this field	<p>Specify</p> <p>Must Understand determines the <code>mustUnderstand</code> attribute of the WS-Addressing headers.</p> <table border="1"> <thead> <tr> <th>Select</th> <th>To</th> </tr> </thead> <tbody> <tr> <td>True</td> <td> <p>Indicate that processing the WS-Addressing headers is required by the recipients.</p> <p>If you select True for Must Understand and the SOAP node receives a header that it does not understand or cannot process, it returns a fault.</p> </td> </tr> <tr> <td>False</td> <td> <p>Indicate that processing the WS-Addressing headers is optional. This is the default.</p> </td> </tr> </tbody> </table> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: In SOAP 1.1, the values of the <code>mustUnderstand</code> attribute were 0 and 1 instead of True and False; however, Integration Server processes both sets of values the same way and performs any necessary conversions.</p> <p>For more information about the <code>mustUnderstand</code> and <code>actor</code> attributes in SOAP 1.1, see the <i>Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000</i>.</p> <p>For more information about the <code>mustUnderstand</code> and <code>role</code> attributes in SOAP 1.2, see the <i>Simple Object Access Protocol (SOAP) 1.2 specification</i>.</p> </div>	Select	To	True	<p>Indicate that processing the WS-Addressing headers is required by the recipients.</p> <p>If you select True for Must Understand and the SOAP node receives a header that it does not understand or cannot process, it returns a fault.</p>	False	<p>Indicate that processing the WS-Addressing headers is optional. This is the default.</p>
Select	To						
True	<p>Indicate that processing the WS-Addressing headers is required by the recipients.</p> <p>If you select True for Must Understand and the SOAP node receives a header that it does not understand or cannot process, it returns a fault.</p>						
False	<p>Indicate that processing the WS-Addressing headers is optional. This is the default.</p>						
Role	<p>Target of the WS-Addressing headers in the SOAP message. Role determines the value of the <code>role</code> attribute for the WS-Addressing headers. The <code>actor</code> or <code>role</code> attribute specifies a URI for the recipient of WS-Addressing header entries.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: In SOAP 1.1, the <code>role</code> attribute is named <code>actor</code>; however, Integration Server processes both names the same and performs any necessary conversions.</p> </div> <table border="1"> <thead> <tr> <th>Select</th> <th>To</th> </tr> </thead> <tbody> <tr> <td>Ultimate Receiver</td> <td> <p>Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.</p> </td> </tr> <tr> <td>Next</td> <td> <p>Specify the following URI for the <code>role</code> attribute:</p> <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next" </td> </tr> </tbody> </table>	Select	To	Ultimate Receiver	<p>Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.</p>	Next	<p>Specify the following URI for the <code>role</code> attribute:</p> <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next"
Select	To						
Ultimate Receiver	<p>Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.</p>						
Next	<p>Specify the following URI for the <code>role</code> attribute:</p> <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next" 						

In this field	Specify <ul style="list-style-type: none">■ For SOAP 1.1: "http://schemas.xmlsoap.org/soap/actor/next"
	None Specify the following URI for the role attribute: <ul style="list-style-type: none">■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/none"■ For SOAP 1.1: "http://www.w3.org/2003/05/soap-envelope/role/none"
	Other Specify the target of the header. Typically, this will be a URI.
From	URI of the source of the SOAP response. <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
ReplyTo	URI to which the response (reply) messages are to be routed. This property is optional. <p>If this value is not specified, the default values for this URI depends on the WS-Addressing policy attached to the web service descriptor.</p> <ul style="list-style-type: none">■ For the Final version of WS-Addressing, ReplyTo defaults to http://www.w3.org/2005/08/addressing/anonymous.■ For the Submission version of WS-Addressing, ReplyTo defaults to http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous. <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the response message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>

In this field	Specify
FaultTo	<p>URI to which the SOAP fault messages are to be routed. This property is optional.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the fault message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>

11. Click **Save Changes**.

Creating an Endpoint Alias for a Provider Web Service Descriptor for Use with JMS

If a provider web service descriptor binder specifies the JMS transport, you must assign a web service endpoint alias to the binder. For a web service descriptor that uses SOAP over JMS, the provider web service endpoint alias provides the following information:

- JMS message header information for the request message, such as delivery mode, time to live, and the destination for replies. Integration Server uses this information to populate the binding elements in the WSDL generated for the web service descriptor.
- The SOAP-JMS trigger that listens for SOAP over JMS messages for the web service descriptor. The SOAP-JMS trigger also provides the JMS connection information needed to create a connection on the JMS provider. Integration Server uses the information provided by the SOAP-JMS trigger to construct most of the JMS URI (the web service descriptor determines the targetService). The JMS URI appears in the WSDL document as the value of the "location=" attribute for the address element within the port element.
- WS Security Properties that specify the information needed by the SOAP processor to decrypt and verify the inbound SOAP request and/or encrypt and sign the outbound SOAP response and the details for adding the timestamp information.

Note:

WS-Security credentials such as private keys and public keys do not always need to be provided in a web service endpoint alias. If this information is not provided in the alias, Integration Server can obtain the information from other locations. For more information

about usage and resolution order of certificates and keys for WS-Security, see the *Web Services Developer's Guide*.

- Message addressing properties that provides addressing information relating to the delivery of a message to a web service. This includes the destination address of a message or fault and the authentication credentials required to send a response to a different address than the one from which the request was received.

Keep the following information in mind when creating a web service endpoint alias for a JMS binder in a provider web service descriptor:

- You can associate the web service endpoint alias with:
 - A SOAP-JMS trigger that already exists.
 - A WS endpoint trigger that you create at the same time you create the endpoint alias.
- If you use a SOAP-JMS trigger in the web service endpoint alias and subsequently assign the alias to a JMS binder in a provider web service descriptor, the web service descriptor has a dependency on the SOAP-JMS trigger. Consequently, at start up or when reloading the package containing the web service descriptor, Integration Server must load the SOAP-JMS trigger before loading the web service descriptor. If the SOAP-JMS trigger and web service descriptor are not in the same package, you need to create a package dependency for the package that contains the web service descriptor on the package that contains the SOAP-JMS trigger.
- If you rename the SOAP-JMS trigger assigned to an alias, you need to update the alias to use the renamed trigger.
- The following properties are optional.
 - **Delivery Mode**
 - **Time to Live**
 - **Priority**
 - **Reply To Name**
 - **Reply To Type**
- If you do not specify values for one of the listed properties (or specify an invalid value), Integration Server will not include information for the property in the WSDL document generated for a provider web service descriptor that uses the web service endpoint alias. The absence of the property from the WSDL document instructs the web service consumer to use the default value for the property as indicated in the Java Message Service standard.

➤ **To create a provider web service endpoint alias for use with JMS**

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.

3. Click **Create Web Service Endpoint Alias**.
4. Under **Web Service Endpoint Alias Properties**, provide the following information:

In this field	Specify
Alias	A name for the JMS provider web service endpoint alias. The alias name cannot include the following illegal characters: <code># © \ & @ ^ ! % * : \$. / \ \ ` ; , ~ + =) (} { } [> < "</code>
Description	A description for the endpoint alias.
Type	Provider
Transport Type	JMS

5. Under **JMS Transport Properties**, provide the following information:

In this field	Specify						
JMS Trigger Name	The name of the SOAP-JMS trigger used to <ul style="list-style-type: none"> ■ Receive JMS messages. ■ Supply JMS connection properties to any web service descriptors using this web service endpoint alias. <p>If you want to use a WS endpoint trigger, select WS Endpoint Trigger. For more information about WS endpoint triggers, see “About WS Endpoint Triggers” on page 875.</p>						
Delivery Mode	The message delivery mode for the request message. This is the delivery mode that web service clients must specify in the JMS message that serves as the request message for the web service.						
	<table border="1"> <thead> <tr> <th>Select</th> <th>To</th> </tr> </thead> <tbody> <tr> <td>PERSISTENT</td> <td>Indicate the request message should be persistent. The message will not be lost if the JMS provider fails.</td> </tr> <tr> <td>NON_PERSISTENT</td> <td>Indicate the request message is not persistent. The message might be lost if the JMS provider fails.</td> </tr> </tbody> </table>	Select	To	PERSISTENT	Indicate the request message should be persistent. The message will not be lost if the JMS provider fails.	NON_PERSISTENT	Indicate the request message is not persistent. The message might be lost if the JMS provider fails.
Select	To						
PERSISTENT	Indicate the request message should be persistent. The message will not be lost if the JMS provider fails.						
NON_PERSISTENT	Indicate the request message is not persistent. The message might be lost if the JMS provider fails.						
Time to Live	The number of milliseconds that can elapse before the request message expires on the JMS provider. A value of 0 indicates that the message does not expire.						

In this field	Specify
Priority	Specifies the message priority. The JMS standard defines priority levels from 0 to 9, with 0 as the lowest priority and 9 as the highest.
Reply To Name	Name or lookup name of the destination to which the web service sends a response (reply) message. Specify a name if the JMS connection alias used by the SOAP-JMS trigger connects to the webMethods Broker natively. Specify a lookup name if the JMS connection alias uses JNDI to retrieve a connection factory that is then used to connect to the JMS provider.
Reply To Type	Type of destination to which the web service sends the response (reply) message. Specify the destination type if the following are true: <ul style="list-style-type: none">■ The web service descriptor to which the endpoint alias is assigned use the In-Out message exchange pattern.■ The JMS connection alias specified by the SOAP-JMS trigger connects to the webMethods Broker natively. On the webMethods Broker, a queue and topic can have the same name. You must specify Reply To Type to indicate to which destination the reply will be sent.

Select	To
QUEUE	Indicate that the web service sends the response message to a particular queue.
TOPIC	Indicate that the web service sends the request message to a particular topic.

6. Under **JMS WSDL Options**, provide the following information:

Select	To
Include Connection Factory Name	Include the connection factory name in the JMS URI.
Include JNDI Parameters	Include the JNDI parameters in the JMS URI.

Note:

The JMS URI appears in the WSDL document as the location attribute value for the address element contained within the port element.

7. Under **WS Security Properties**, if the inbound SOAP request must be decrypted and/or the outbound SOAP response must be signed, do the following:

In this field	Specify
Keystore Alias	Alias of the keystore containing the private key used to decrypt the inbound SOAP request or sign the outbound SOAP response.
	<p>Important: The provider must have already given the consumer the corresponding public key.</p>
Key Alias	Alias of the private key used to decrypt the request or sign the response. The key must be in the keystore specified in Keystore Alias .

8. Under **WS Security Properties**, if the signing certificate chain of an inbound signed SOAP message has to be *verified*, specify the following:

In this field	Specify
Truststore Alias	The alias for the truststore that contains the list of CA certificates that Integration Server uses to validate the trust relationship.

9. Under **WS Security Properties**, configure how Integration Server handles timestamps in the security header.

In this field	Specify
Timestamp Precision	<p>Whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss:SSS'Z'</code>. If you set the precision to seconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss'Z'</code>.</p> <p>If you do not select a precision value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.</p>
Timestamp Time to Live	<p>The time-to-live value for the outbound message in seconds. Integration Server uses the time-to-live value to set the expiry time in the Timestamp element of outbound messages. The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a Timestamp Time to Live value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p> <p>Note: The Timestamp Time to Live value should be greater than the Time to Live value specified under JMS Transport Properties.</p>

In this field	Specify
Timestamp Maximum Skew	<p>The maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed. Specify a positive integer or zero.</p> <p>Integration Server uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. Integration Server validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>

10. Under **Message Addressing Properties**, provide the following addressing information relating to the delivery of the message. The message addressing properties define the addressing information that can be attached to the SOAP message.

In this field	Specify
To	<p>RI of the destination of the SOAP message.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <code><wsa:ReferenceParameters></code> properties of the endpoint reference to which the message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
Response Map	<p>Address to which the provider will send the reply or fault message and the corresponding message addressing alias. Integration Server retrieves the authentication details needed to send the response from the message addressing alias mapped to the address.</p> <p>In the Address field, specify the URI to which the provider will send the reply or the fault message.</p> <p>From the Message Addressing Alias list, select the Message Addressing endpoint alias from which Integration Server will retrieve the authentication details. Integration Server uses the authentication details to send the response to the ReplyTo or FaultTo endpoints.</p> <p>Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>

11. Click **Save Changes**.

If you selected WS endpoint trigger for **JMS Trigger Name**, saving the alias creates the WS endpoint trigger. Before the WS endpoint trigger is usable, you will need to specify a destination and enable it. You can also change the values of some message processing properties. For more information about editing a WS endpoint trigger, see [“Editing WS Endpoint Triggers” on page 876](#).

Note:

If a provider web service endpoint alias for use with JMS specifies a WS endpoint trigger, deleting the alias also deletes the WS endpoint trigger.

Creating an Endpoint Alias for a Consumer Web Service Descriptor for Use with JMS

A web service endpoint alias for use with a consumer web service descriptor that has a JMS binder specifies how and where Integration Server sends a request message when executing a web service descriptor.

When creating a consumer web service descriptor, Integration Server extracts the JMS information from the WSDL document and saves it with the binder information in the web service descriptor. However, as indicated in the SOAP over Java Message Service standard, the only JMS information required in the WSDL is the lookup variant and the destination name. Consequently, it is possible that some information necessary to connect to the JMS provider is absent from the WSDL. Integration Server uses the information in a JMS consumer web service endpoint alias to replace or supplement the JMS information specified in the WSDL document.

When creating a consumer web service descriptor, the message addressing properties define the WS-addressing headers information that can be attached to the SOAP message.

Keep the following points in mind when creating a web service endpoint alias for use with a consumer web service descriptor with a SOAP over JMS binding:

- A JMS consumer web service endpoint alias can specify one of the following options to connect to a JMS provider:
 - JNDI provider alias and a connection factory.
 - JMS connection alias.

Only specify a JNDI provider alias and connection factory, or JMS connection alias, if information for connecting to the JMS provider was not included in the WSDL document used to create the consumer web service descriptor or if you want to overwrite the connection information included in the WSDL document.

Note:

Using a JMS connection alias to connect to the JMS provider might offer better performance. Keep in mind that a JMS connection alias can connect to the JMS provider by using JNDI to retrieve a connection factory and then establishing a connection or by connecting natively to the webMethods Broker.

- If you want to use the client side queue with the web service descriptor to which the alias is assigned, you must specify a JMS connection alias as the way to connect to the JMS provider.
- Information in the JMS consumer web service endpoint alias can supplement or replace the JMS URI information obtained from a WSDL.
- You can use the endpoint alias to provide information for the WS-Security header as determined by the security policy for the web service. A web service security policy can require that:
 - SOAP message requests include a Username token.
 - SOAP message response be decrypted.
 - SOAP message requests to be signed.
 - X.509 authentication.
 - A Timestamp element be added to the security header.

Note:

WS-Security credentials such as private keys and public keys do not always need to be provided in a web service endpoint alias. If this information is not provided in the alias, Integration Server can obtain the information from other locations. For more information about usage and resolution order of certificates and keys for WS-Security, see the *Web Services Developer's Guide*.

➤ **To create a consumer web service endpoint alias for use with JMS**

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Create Web Service Endpoint Alias**.
4. Under **Web Service Endpoint Alias Properties**, provide the following information:

In this field	Specify
Alias	A name for the JMS consumer web service endpoint alias. The alias name cannot include the following illegal characters: # © \ & @ ^ ! % * : \$. / \ \ ` ; , ~ + =) (} {] [> < "
Description	A description for the endpoint alias.
Type	Consumer
Transport Type	JMS

In this field	Specify
Execute ACL	ACL that governs which user groups on your server can use this web service endpoint alias. Select an ACL from the drop down list. By default, only members of groups governed by the Internal ACL can use this alias.

5. Under **JMS Transport Properties**, do the following if you want to connect to the JMS provider using a connection factory:

In this field	Specify
Connect Using	JNDI Properties
JNDI Provider Alias	The alias for the JNDI provider that Integration Server uses to look up administered objects. For information about creating a JNDI provider alias, see “Creating a JNDI Provider Alias” on page 275 .
Connection Factory Name	The lookup name for the connection factory to use to create a connection to the JMS provider.

Note:

You need to specify a connection factory only if the WSDL document used to create the consumer web service descriptor did not specify a connection factory or you want to overwrite the connection factory.

6. Under **JMS Transport Properties**, do the following if you want to connect to the JMS provider using a JMS connection alias:

In this field	Specify
Connect Using	JMS Connection Alias
JMS Connection Alias	The name of the JMS connection alias that you want Integration Server to use to connect to the JMS provider. For information about creating a JMS connection alias, see “Creating a JMS Connection Alias” on page 285 .

7. Under **WS Security Properties**, provide the following information if the WS-Security policy for this consumer web service descriptor requires that SOAP message requests include a UsernameToken.

In this field	Specify
User Name	The user name to include with the UsernameToken.
Password	The password to include with the UsernameToken (must be plain text).

In this field	Specify
---------------	---------

Retype Password	Re-enter the above password.
------------------------	------------------------------

8. If the security policy (or policies) that will be used by this web service requires its requests to be *signed*, requires an X.509 authentication token to be included, or requires that SOAP message responses be encrypted, specify the following:

In this field	Specify
---------------	---------

Keystore Alias	Alias to the keystore that contains the private key used to: <ul style="list-style-type: none">■ Sign outbound SOAP requests■ Include an X.509 authentication token for outbound SOAP requests■ Decrypt inbound SOAP responses
-----------------------	--

Important:

To verify messages from this consumer, the web services provider must have a copy of the corresponding public key.

Key Alias	Alias to the private key used to sign and/or include X.509 authentication token for outbound SOAP messages and/or decrypt inbound SOAP responses. The key must be in the keystore specified in Keystore Alias .
------------------	--

9. Under **WS Security Properties**, specify the provider's certificate file. This certificate is used to encrypt the outbound SOAP request and/or verify the inbound SOAP response.

In this field	Specify
---------------	---------

Partner's Certificate	The path and file name of the provider's certificate, which contains its public key.
------------------------------	--

10. Under **WS Security Properties**, if the security policy (or policies) that will be used by this web services consumer requires that responses be *verified* by a trusted authority, specify the following:

In this field	Specify
---------------	---------

Partner's Certificate	Path and file name of the file containing the provider's certificate.
------------------------------	---

Truststore Alias	The alias for the truststore that contains the list of CA certificates that Integration Server uses to validate the trust relationship.
-------------------------	---

11. Under **WS Security Properties**, configure how Integration Server handles timestamps in the security headers.

In this field	Specify
Timestamp Precision	<p>Whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss:SSS'Z'</code>. If you set the precision to seconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss'Z'</code>.</p> <p>If you do not select a precision value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.</p>
Timestamp Time to Live	<p>The time-to-live value for the outbound message in seconds. Integration Server uses the Timestamp Time to Live value to set the expiry time in the Timestamp element of outbound messages. The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a time-to-live value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p>
Timestamp Maximum Skew	<p>The maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed. Specify a positive integer or zero.</p> <p>Integration Server uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. Integration Server validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>

For more information about timestamps in the WS-Security header, see [“Timestamps in the WS-Security Header” on page 392](#).

- Under **Message Addressing Properties**, provide the following addressing information relating to the delivery of a message to a web service.

In this field	Specify
Must Understand	<p>Whether the recipients (the actor or role to which the header is targeted) are required to process the WS-Addressing headers. Recipients that cannot process a mandatory WS-Addressing header reject the message and return a SOAP fault.</p> <p>Must Understand determines the <code>mustUnderstand</code> attribute of the WS-Addressing headers.</p>

In this field	Specify
Select	To
True	Indicate that processing the WS-Addressing headers is required by the recipients (the actor or role to which the header is targeted). If you select True for Must Understand and the SOAP node receives a header that it does not understand or cannot process, it returns a fault.
False	Indicate that processing the WS-Addressing headers is optional. This is the default.

Note:

In SOAP 1.1, the values of the mustUnderstand attribute were 0 and 1 instead of True and False; however, Integration Server processes both sets of values the same and performs any necessary conversions.

For more information about the mustUnderstand and actor attributes in SOAP 1.1, see the *Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000*.

For more information about the mustUnderstand and role attributes in SOAP 1.2, see the *Simple Object Access Protocol (SOAP) 1.2 specification*.

Role	Target of the WS-Addressing headers in the SOAP message. Role determines the value of the role attribute for the WS-Addressing headers. The actor or role attribute specifies a URI for the recipient of WS-Addressing header entries.
-------------	--

Note:

In SOAP 1.1, the role attribute is named actor; however, Integration Server processes both names the same and performs any necessary conversions.

Select	To
Ultimate Receiver	Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.
Next	Specify the following URI for the role attribute: <ul style="list-style-type: none">■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next"■ For SOAP 1.1: "http://schemas.xmlsoap.org/soap/actor/next"

In this field	<p>Specify</p> <p>None Specify the following URI for the role attribute:</p> <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/none" ■ For SOAP 1.1: "http://www.w3.org/2003/05/soap-envelope/role/none" <p>Other Specify the target of the header. Typically, this will be a URI.</p>
To	<p>URI of the destination of the SOAP request.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the request is addressed. You can specify more than one reference parameter. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
From	<p>URI of the source of the SOAP message.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
ReplyTo	<p>URI to which the response (reply) messages are to be routed. This property is optional.</p> <p>If this value is not specified, the default values for this URI depends on the WS-Addressing policy attached to the web service descriptor.</p> <ul style="list-style-type: none"> ■ For the Final version of WS-Addressing, ReplyTo defaults to http://www.w3.org/2005/08/addressing/anonymous. ■ For the Submission version of WS-Addressing, ReplyTo defaults to http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous. <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the response message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also</p>

In this field	Specify specify Extensible Elements , which are elements other than those specified as part of the Metadata and Reference Parameters . You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.
FaultTo	URI to which the SOAP fault messages are to be routed. This property is optional. In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the fault message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements , which are elements other than those specified as part of the Metadata and Reference Parameters . You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.

13. Click **Save Changes**.

Creating an Endpoint Alias for Message Addressing for Use with JMS

A web service endpoint alias for message addressing for use with a web service descriptor that has a JMS binder specifies the addressing information relating to the delivery of a SOAP response to the receiver.

Keep the following points in mind when creating a web service endpoint alias for message addressing for use with a web service descriptor with a SOAP over JMS binding:

- A JMS message addressing web service endpoint alias can specify one of the following options to connect to a JMS provider:
 - JNDI provider alias and a connection factory.
 - JMS connection alias.

Only specify a JNDI provider alias and connection factory, or JMS connection alias, if information for connecting to the JMS provider was not included in the WSDL document used to create the consumer web service descriptor or if you want to overwrite the connection information included in the WSDL document.

Note:

Using a JMS connection alias to connect to the JMS provider might offer better performance. Keep in mind that a JMS connection alias can connect to the JMS provider by using JNDI to retrieve a connection factory and then establishing a connection or by connecting natively to the webMethods Broker.

- You can use the endpoint alias to provide information for the WS-Security header as determined by the security policy for the web service.

Note:

WS-Security credentials such as private keys and public keys do not always need to be provided in a message addressing web service endpoint alias. If this information is not provided in the alias, Integration Server can obtain the information from other locations. For more information about usage and resolution order of certificates and keys for WS-Security, see the *Web Services Developer's Guide*.

- If you want to use the client side queue with the web service descriptor to which the alias is assigned, you must specify a JMS connection alias as the way to connect to the JMS provider.

Note:

You cannot delete a message addressing endpoint alias if a web service endpoint alias for provider web service descriptor is using the message addressing endpoint alias as a part of its response map.

➤ To create a message addressing web service endpoint alias for use with JMS

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Create Web Service Endpoint Alias**.
4. Under **Web Service Endpoint Alias Properties**, provide the following information:

In this field	Specify
Alias	A name for the JMS message addressing web service endpoint alias. The alias name cannot include the following illegal characters: # © \ & @ ^ ! % * : \$. / \ \ ` ; , ~ + =) (} { } [> < "
Description	A description for the endpoint alias.
Type	Message Addressing
Transport Type	JMS

5. Under **JMS Transport Properties**, do the following if you want to connect to the JMS provider using a connection factory:

In this field	Specify
Connect Using	JNDI Properties
JNDI Provider Alias	The alias for the JNDI provider that Integration Server uses to look up administered objects. For information about creating a JNDI provider alias, see “Creating a JNDI Provider Alias” on page 275 .
Connection Factory Name	The lookup name for the connection factory to use to create a connection to the JMS provider.

6. Under **JMS Transport Properties**, do the following if you want to connect to the JMS provider using a JMS connection alias:

In this field	Specify
Connect Using	JMS Connection Alias
JMS Connection Alias	The name of the JMS connection alias that you want Integration Server to use to connect to the JMS provider. For information about creating a JMS connection alias, see “Creating a JMS Connection Alias” on page 285 .

7. Under **WS Security Properties**, specify the certificate file of the receiver of the SOAP response. This certificate is used to encrypt the outbound SOAP response.

In this field	Specify
Partner's Certificate	The path and file name of the certificate file of the receiver of the SOAP response, which contains its public key.

8. Under **WS Security Properties**, specify the following if the security policy (or policies) that will be used by this web service requires its responses to be *signed*, requires an X.509 authentication token to be included, or requires that SOAP message responses be encrypted.

In this field	Specify
Keystore Alias	Alias to the keystore that contains the private key used to: <ul style="list-style-type: none">■ Sign outbound SOAP responses■ Include an X.509 authentication token for outbound SOAP responses

Important:

In this field	Specify
	To verify response messages from this web service, the receiver must have the corresponding public key.
Key Alias	Alias to the private key used to sign and/or include X.509 authentication token for outbound SOAP messages. The key must be in the keystore specified in Keystore Alias .

9. Under **WS Security Properties**, configure how Integration Server handles timestamps in the security headers.

In this field	Specify
Timestamp Precision	<p>Whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss:SSS'Z'</code>. If you set the precision to seconds, Integration Server uses the timestamp format <code>yyyy-MM-dd'T'HH:mm:ss'Z'</code>.</p> <p>If you do not select a precision value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.</p>
Timestamp Time to Live	<p>The time-to-live value for the outbound message in seconds. Integration Server uses the Timestamp Time to Live value to set the expiry time in the Timestamp element of outbound messages. The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a time-to-live value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p>
Timestamp Maximum Skew	<p>The maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed. Specify a positive integer or zero.</p> <p>Integration Server uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. Integration Server validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, Integration Server will use the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>

For more information about timestamps in the WS-Security header, see [“Timestamps in the WS-Security Header” on page 392](#).

10. Under **Message Addressing Properties**, provide the following addressing information relating to the delivery of a response SOAP message to the receiver. The message addressing properties define the addressing information that can be attached to the SOAP message.

In this field **Specify**

Must Understand Whether the recipients (the actor or role to which the header is targeted) are required to process the WS-Addressing headers. Recipients that cannot process a mandatory WS-Addressing header reject the message and return a SOAP fault.

Must Understand determines the `mustUnderstand` attribute of the WS-Addressing headers.

Select	To
--------	----

True	Indicate that processing the WS-Addressing headers is required by the recipients.
-------------	---

If you select **True** for **Must Understand** and the SOAP node receives a header that it does not understand or cannot process, it returns a fault.

False	Indicate that processing the WS-Addressing headers is optional. This is the default.
--------------	--

Note:

In SOAP 1.1, the values of the `mustUnderstand` attribute were 0 and 1 instead of True and False; however, Integration Server processes both sets of values the same way and performs any necessary conversions.

For more information about the `mustUnderstand` and actor attributes in SOAP 1.1, see the *Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000*.

For more information about the `mustUnderstand` and role attributes in SOAP 1.2, see the *Simple Object Access Protocol (SOAP) 1.2 specification*.

Role Target of the WS-Addressing headers in the SOAP message. Role determines the value of the role attribute for the WS-Addressing headers. The actor or role attribute specifies a URI for the recipient of WS-Addressing header entries.

Note:

In SOAP 1.1, the role attribute is named actor; however, Integration Server processes both names the same and performs any necessary conversions.

In this field	Specify	To
	Select	
	Ultimate Receiver	Indicate that the recipient is the ultimate destination of the SOAP message. This is the default.
	Next	Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/next" ■ For SOAP 1.1: "http://schemas.xmlsoap.org/soap/actor/next"
	None	Specify the following URI for the role attribute: <ul style="list-style-type: none"> ■ For SOAP 1.2: "http://www.w3.org/2003/05/soap-envelope/role/none" ■ For SOAP 1.1: "http://www.w3.org/2003/05/soap-envelope/role/none"
	Other	Specify the target of the header. Typically, this will be a URI.
From	URI of the source of the SOAP response.	
	<p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <code><wsa:ReferenceParameters></code> properties of the endpoint reference. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>	
ReplyTo	URI to which the response (reply) messages are to be routed. This property is optional.	
	If this value is not specified, the default values for this URI depends on the WS-Addressing policy attached to the web service descriptor.	
	<ul style="list-style-type: none"> ■ For the Final version of WS-Addressing, ReplyTo defaults to <code>http://www.w3.org/2005/08/addressing/anonymous</code>. 	
	<ul style="list-style-type: none"> ■ For the Submission version of WS-Addressing, ReplyTo defaults to <code>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</code>. 	
	In the Reference Parameters field, specify additional parameters, if any, that correspond to <code><wsa:ReferenceParameters></code> properties of	

In this field	Specify <p>the endpoint reference to which the response message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>
FaultTo	<p>URI to which the SOAP fault messages are to be routed. This property is optional.</p> <p>In the Reference Parameters field, specify additional parameters, if any, that correspond to <wsa:ReferenceParameters> properties of the endpoint reference to which the fault message is addressed. Optionally, you can specify metadata (such as WSDL or WS-Policy) about the service in the Metadata Elements field. You can also specify Extensible Elements, which are elements other than those specified as part of the Metadata and Reference Parameters. You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p> <p>You can specify more than one reference parameter, metadata element, or extensible element. Click the '+' icon to add more rows and the 'x' icon to delete the rows.</p>

11. Click **Save Changes**.

Timestamps in the WS-Security Header

The WS-Security header can contain Timestamp elements and tokens. Integration Server uses the timestamp to specify or detect whether an outbound or inbound message expires, specifically:

- For outbound messages, if the WS-Security policy attached to the web service descriptor includes the <sp:IncludeTimestamp/> assertion, Integration Server adds a Timestamp element, which includes the creation and expiry time, to the security header.
- For inbound messages, if the message has a Timestamp token, based on the Timestamp token, Integration Server verifies that the message has not arrived after the expiration time.

In the web service endpoint alias, you can specify the precision of the message timestamp, the message time to live, and whether to account for any difference in the clocks on the sending and receiving machines.

17 Configuring Reliable Messaging in Integration Server

■ Overview of Reliable Messaging	394
■ Using Reliable Messaging in Integration Server	395
■ Configuring Reliable Messaging in Integration Server	397
■ Reliable Messaging Sequence Reports	399
■ Closing a Sequence	401
■ Terminating a Sequence	401
■ Sending an Acknowledgement Request	402

Overview of Reliable Messaging

Integration Server uses the Web Services Reliable Messaging (WS-ReliableMessaging) protocol to reliably send SOAP messages between web service providers and consumers. Reliable transmission of messages ensures that the SOAP messages are delivered even if the destination endpoint is temporarily unavailable or if the network connection fails.

The following steps describe what happens during exchange of SOAP messages where reliable messaging is used:

1. The reliable messaging source sends a message or a series of messages that are transmitted across a communication link to a reliable messaging destination. Along with the messages, the reliable messaging source also sends a request to the recipient to acknowledge the messages.
2. When the reliable messaging destination receives the messages, it sends an acknowledgement back to the reliable messaging source either individually for each message or as a single acknowledgement for a series of messages.
3. If the messages are not delivered in the first attempt, the reliable messaging source retransmits the messages based on the reliable messaging configuration until the message is delivered and the acknowledgement is received or the sequence has timed out or is terminated.

Understanding Reliable Messaging Terminology

Before configuring reliable messaging in Integration Server, you may find it helpful to first understand the following terminology:

- **Reliable messaging source.** The web service endpoint that transmits the SOAP messages to a reliable messaging destination. In the case of Integration Server, a consumer web service descriptor sending a request or a provider web service descriptor sending a response is the reliable messaging source.
- **Reliable messaging destination.** The web service endpoint that receives SOAP messages that were reliably transmitted from a reliable messaging source. In the case of Integration Server, a consumer web service descriptor receiving a response or a provider web service descriptor receiving a request is the reliable messaging destination.
- **Reliable messaging server.** Integration Server acting as a web service provider in a reliable messaging scenario.
- **Reliable messaging client.** Integration Server acting as a web service consumer in a reliable messaging scenario.
- **Message sequence.** A message or a series of messages that have the same destination. Grouping of messages to a message sequence makes it easy to track and manage the transmission of messages.
- **Sequence key.** User-defined key to identify a message sequence. A reliable messaging client associates a sequence key to a message sequence based on the endpoint URL to which the message sequence is directed. In cases where there are several message sequences directed to the same endpoint URL, you can specify a custom sequence key to identify each sequence.

Each sequence is then uniquely identified by the endpoint URL and the user-specified sequence key. The sequence key can be provided as the input of the `pub.soap.wsrn:createSequence` service or as the input parameter of the web service connector.

- **Client sequence Id.** Unique identifier that is generated by a reliable messaging client and associated with a reliable messaging sequence. To ensure that a client can identify the message as part of a particular sequence, the reliable messaging server specifies the client sequence Id when it sends a SOAP response to the reliable messaging client.
- **Server sequence Id.** Unique identifier that is generated by a reliable messaging server and associated with a reliable messaging sequence. To ensure that a server can identify the message as part of a particular sequence, the reliable messaging client specifies the server sequence Id while sending a SOAP request to the reliable messaging server. The server sequence Id of a message sequence is returned as the output parameter of the `pub.soap.wsrn:createSequence` service or as the `reliableMessagingInfo/responseReliableMessagingProperties/serverSequenceId` output parameter of the web service connector.
- **Acknowledgement.** Response from the reliable messaging destination to the reliable messaging source indicating that the message has been successfully received by the destination.

Using Reliable Messaging in Integration Server

Keep the following points in mind when you configure Integration Server to use reliable messaging:

- Integration Server supports WS-ReliableMessaging Version 1.1 only.
- The namespace prefix `wsrn` represents the URI `http://docs.oasis-open.org/ws-rx/wsrn/200702`.
- The namespace prefix `wsrmp` represents the URI `http://docs.oasis-open.org/ws-rx/wsrmp/200702`.
- To use WS-ReliableMessaging, you attach a standard WS-Policy, which includes reliable messaging assertions, to a web service descriptor. For more information about defining your own policies, see *Web Services Developer's Guide*. For instructions about attaching a WS-Policy to a web service descriptor, see *webMethods Service Development Help*.
- To use WS-ReliableMessaging, you can attach a pre-defined WS-Policy named `ReliableMessaging` that Integration Server provides. This policy is available in the following directory:


```
Integration Server_directory \instances\instance_name\config\wss\policies
```
- Integration Server uses a unique reliable messaging sequence key to track the progress of a set of messages that are exchanged reliably between a web service provider and consumer.
- Integration Server supports only the In Order delivery assurance for reliable messaging.
- You use the Integration Server Administrator to configure reliable messaging for web services. By default, Integration Server applies the reliable messaging configuration defined on the **Settings > Web Services > Reliable Messaging > Edit Configuration** page to all web service providers and consumers. If you want to override the server-level reliable messaging

configuration for a specific web service provider or consumer, define reliable messaging properties for the associated web service endpoint alias.

- Integration Server provides `pub.soap.wsm` built-in services to create and manage reliable messaging sequences. For more information about these built-in services, see *webMethods Integration Server Built-In Services Reference*.

Persistent Storage Support for Reliable Messaging Data

Integration Server provides persistent storage capability for reliable messaging transactional data. Persistent storage support ensures that the messages that are being exchanged between a reliable messaging source and a reliable messaging destination are not lost in case of system or communication failures. When messages are exchanged in distributed systems, errors can occur during the transmission of messages over communication links or during the processing of messages in system components. Under these conditions, Integration Server ensures that no messages are lost and that messages can be eventually recovered after system failure.

Integration Server provides support for persistent storage of information related to reliable messaging sequences, including the essential routing and delivery information.

Keep the following points in mind while configuring Integration Server to provide persistent storage capability for reliable messaging:

- The `ISInternal` functional alias (specified on the **Settings > JDBC Pools** screen) must be configured to point to either the embedded IS Internal database or to the external RDBMS that Integration Server must use for persistent storage.
- If the Integration Servers used for reliable message exchanges are in a clustered environment and are connected to the same database, all the Integration Servers must have the same persistence configuration defined on the **Settings > Web Services > Reliable Messaging > Edit Configuration** page.
- Integration Server supports reliable messaging in a clustered environment only if you configure Integration Server to provide persistent storage capability for reliable messaging.
- For the authentication details to be persisted across Integration Server restarts, you must provide the authentication details for the consumer web service descriptor in the associated consumer endpoint alias and not in the associated connector signature.

Limitations When Using Reliable Messaging in Integration Server

- Integration Server supports only WS-ReliableMessaging Version 1.1.
- Integration Server does not support reliable messaging over JMS.
- Because of interoperability issues with WS-Security and WS-ReliableMessaging standards, Integration Server does not support reliable messaging if the attached reliable messaging policy contains both WS-ReliableMessaging and WS-Security policy assertions.
- Integration Server does not support the following WS-ReliableMessaging assertions:

- `wsrmp:SequenceSTR`
- `wsrmp:SequenceTransportSecurity`

Configuring Reliable Messaging in Integration Server

Use Integration Server Administrator to configure reliable messaging for web services.

> To configure reliable messaging properties

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Reliable Messaging**.
4. Click **Edit Reliable Messaging Configuration**.
5. Under **Reliable Messaging Properties**, provide the following information:

In this field	Specify
Retransmission Interval	The time interval (in milliseconds) for which a reliable messaging source waits for an acknowledgement from a reliable messaging destination before retransmitting the SOAP message. The default is 6000 milliseconds.
Acknowledgement Interval	<p>The time interval (in milliseconds) for which a reliable messaging destination waits before sending an acknowledgement for a message sequence. Messages of the same sequence received within the specified acknowledgement interval are acknowledged in one batch. If there are no other messages to be sent to the acknowledgement endpoint within the time specified as the acknowledgement interval, the acknowledgement is sent as a stand-alone message.</p> <p>The default is 3000 milliseconds.</p>
Exponential Backoff	Whether to use the exponential backoff algorithm to adjust the retransmission interval of unacknowledged messages. Adjusting the time interval between retransmission attempts ensures that a reliable messaging destination does not get flooded with a large number of retransmitted messages.
Select	To
true	Increase the successive retransmission intervals exponentially, based on the specified

In this field	Specify						
	<p>retransmission interval. For example, if the specified retransmission interval is 2 seconds, and the exponential backoff value is set to true, successive retransmission intervals will be 2, 4, 8, 16, 32, and so on if messages continue to be unacknowledged. This is the default.</p> <p>false Use the same time interval specified in the Retransmission Interval field for all retransmissions.</p>						
Inactivity Timeout	<p>The length of time for which a reliable messaging source waits for an acknowledgement from a reliable messaging destination before the source stops retransmitting the SOAP message. Specify the unit of measurement as seconds, minutes, hours, or days.</p> <p>If the reliable messaging source does not receive an acknowledgement within the inactivity timeout specified, it marks the sequence as timed out. You cannot use a sequence if it is timed out. To specify that there is no inactivity timeout limit, set the value of Inactivity Timeout to -1.</p> <p>The default is 60 seconds.</p>						
Sequence Removal Timeout	<p>The length of time for which a reliable messaging source waits after a sequence is terminated before removing the sequence state from memory. Specify the unit of measurement as seconds, minutes, hours, or days. The default is 600 seconds.</p>						
In-Order Delivery Assurance	<p>Whether the messages in a sequence must be delivered to a reliable messaging destination in the same order in which they were sent by the reliable messaging source.</p> <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left; border-bottom: 1px solid black;">Select</th> <th style="text-align: left; border-bottom: 1px solid black;">To</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">True</td> <td>Specify that the messages in a sequence must be delivered to the destination in the same order in which they were sent. This is the default.</td> </tr> <tr> <td style="vertical-align: top;">False</td> <td>Not enforce the delivery of messages in the same order in which were sent.</td> </tr> </tbody> </table>	Select	To	True	Specify that the messages in a sequence must be delivered to the destination in the same order in which they were sent. This is the default.	False	Not enforce the delivery of messages in the same order in which were sent.
Select	To						
True	Specify that the messages in a sequence must be delivered to the destination in the same order in which they were sent. This is the default.						
False	Not enforce the delivery of messages in the same order in which were sent.						
Maximum Retransmission Count	<p>The number of times a reliable messaging source must retransmit a message if an acknowledgement is not received from the reliable messaging destination. To specify that there is no limit to the number of retransmission attempts, set the</p>						

In this field	Specify						
	value of Maximum Retransmission Count to -1. The default is 10.						
Storage Type	Specifies whether Integration Server uses the persistent or non-persistent mode to store the reliable messaging sequence information.						
	<table border="0"> <thead> <tr> <th style="text-align: left; border-bottom: 1px solid black;">Select</th> <th style="text-align: left; border-bottom: 1px solid black;">To</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">Non-Persistent</td> <td>Use a non-persistent mode of storage of reliable messaging sequence information. When the Non-Persistent mode of storage is used, Integration Server relies on the on-heap memory for reliable messaging data storage. When Integration Server restarts, the reliable messaging information will be removed from memory. This is the default.</td> </tr> <tr> <td style="vertical-align: top;">Database</td> <td>Use a persistent mode of storage of reliable messaging information. When the Database mode of storage is used, Integration Server uses a database to store the reliable messaging information. All information related to reliable messaging sequences, including the essential routing and delivery information, is preserved across Integration Server restarts.</td> </tr> </tbody> </table>	Select	To	Non-Persistent	Use a non-persistent mode of storage of reliable messaging sequence information. When the Non-Persistent mode of storage is used, Integration Server relies on the on-heap memory for reliable messaging data storage. When Integration Server restarts, the reliable messaging information will be removed from memory. This is the default.	Database	Use a persistent mode of storage of reliable messaging information. When the Database mode of storage is used, Integration Server uses a database to store the reliable messaging information. All information related to reliable messaging sequences, including the essential routing and delivery information, is preserved across Integration Server restarts.
Select	To						
Non-Persistent	Use a non-persistent mode of storage of reliable messaging sequence information. When the Non-Persistent mode of storage is used, Integration Server relies on the on-heap memory for reliable messaging data storage. When Integration Server restarts, the reliable messaging information will be removed from memory. This is the default.						
Database	Use a persistent mode of storage of reliable messaging information. When the Database mode of storage is used, Integration Server uses a database to store the reliable messaging information. All information related to reliable messaging sequences, including the essential routing and delivery information, is preserved across Integration Server restarts.						
Housekeeping Interval	The time interval (in seconds) in which Integration Server sweeps the database to check for timed-out or terminated sequences.						
	The messages are timed out or terminated depending on the specified Inactivity Timeout and Sequence Removal Timeout values. Integration Server sweeps the database periodically based on the Housekeeping Interval and identifies and marks the messages that are timed out and removes the terminated messages.						
	The default is 20 seconds.						

6. Click **Save Changes**.

Reliable Messaging Sequence Reports

A sequence report provides information about the reliable message sequences that Integration Server is processing or has completed processing. This report includes the client and server sequences, that is, the reliable message sequences that Integration Server as a reliable messaging

client has sent or received and that Integration Server as a reliable messaging server has received or sent.

Client and Server Sequences

Field	Description
User Sequence Key	Unique key identifying a message sequence. The value of User Sequence Key is the same as the value specified in the <i>sequenceKey</i> input parameter of the <code>pub.soap.wsm:createSequence</code> service or as the <i>reliableMessagingProperties/sequenceKey</i> parameter of the web service connector.
Client Sequence Id	Unique identifier that is generated by the reliable messaging client and associated with a reliable messaging sequence.
Server Sequence Id	Unique identifier that is generated by the reliable messaging server and associated with a reliable messaging sequence. This Id is returned by Integration Server as the <i>serverSequenceId</i> output parameter of the <code>pub.soap.wsm:createSequence</code> service or the <i>reliableMessagingInfo/responseReliableMessagingProperties/serverSequenceId</i> output parameter of the web service connector.
To Address	The endpoint address with which the reliable messaging sequence is established.
Status	Status of the message sequence.
Number of Completed Messages	Count of messages in the message sequence that have been delivered and acknowledgement messages have been received.
Last Activated	Timestamp that specifies the time when the last message in the sequence was transmitted.
Acknowledgement Request	Sends an acknowledgement request to the reliable messaging server. For more information about sending a request for acknowledgement, see “Sending an Acknowledgement Request” on page 402 .
Close	Closes the sequence. For more information about closing a sequence, see “Closing a Sequence” on page 401 .
Terminate	Terminates the sequence. For more information about terminating a sequence, see “Terminating a Sequence” on page 401 .

Viewing Reliable Messaging Sequence Reports

You can view details about message sequences in the Reliable Messaging screen in Integration Server Administrator.

➤ To view the reliable messaging sequence report

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Reliable Messaging**.

Integration Server displays the sequence reports under **Client Sequences** and **Server Sequences**.

Closing a Sequence

You can close a message sequence if you do not want a reliable messaging client to send any new messages using the specified sequence, other than those already accepted and being processed by the reliable messaging server.

When a reliable messaging sequence is closed, the reliable messaging server will not accept any new messages of the same sequence. However, the reliable messaging server will continue to track the closed sequence and respond to acknowledgment requests for messages in the closed sequence.

Only a reliable messaging client can close a message sequence. After a reliable messaging sequence is closed, only a client can terminate the sequence. If there is no activity related to the closed sequence over the specified inactivity timeout interval, the reliable messaging client marks the sequence as timed out.

➤ To close a sequence

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Reliable Messaging**.
4. Under **Client Sequences**, locate the sequence that you want to close and click the **Close** link.

Terminating a Sequence

You can terminate a reliable messaging sequence without waiting for all the messages in the sequence to be acknowledged. Only a reliable messaging client can terminate a message sequence. A terminate sequence request from a reliable messaging client indicates that the sequence is complete and that it will not be sending any further messages related to the sequence. You can, however, reuse the sequence key of the message sequence only after the **Sequence Removal Timeout** period has elapsed.

➤ To terminate a sequence

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Reliable Messaging**.
4. Under **Client Sequences**, locate the sequence that you want to terminate and click the **Terminate** link.

Sending an Acknowledgement Request

You can direct the reliable messaging client to request the reliable messaging server to acknowledge a specific message sequence.

> To send an acknowledgement request

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Web Services**.
3. Click **Reliable Messaging**.
4. Under **Client Sequences**, locate the sequence for which you want to send an acknowledgement request to the reliable messaging server. Under **Acknowledgement Request**, click the **Send** link.

18 Configuring Integration Server to Use JWT

■ Overview of JWT	404
■ Using JWT with Integration Server	405
■ Support for JWT in Integration Server	406
■ Configuring Integration Server to use JWT	407

Overview of JWT

JSON Web Token (JWT) is a JSON-based standard (RFC 7519) that provides a secured means of exchanging claims between two parties. JWT represents claims in a compact format, which is intended for space constrained environments such as HTTP authorization headers. JWT encodes claims as JSON objects, which is used as the payload in a JWT.

JWT authentication is stateless as the user state is never saved in the server memory. JWTs are self-contained, that is, it contains all the necessary information about the user.

JWT Structure

The JWT structure is comprised of the following elements:

- **Header.** Identifies the token type and the hashing algorithm used. In this case the token type is JWT.
- **Payload.** Contains the JWT claims. Claims are statements about an entity (user) and the additional metadata that a user wants to transmit to server. Following are the types of claims:
 - **Registered claims.** Set of predefined claims that are not mandatory but recommended.
 - **Public claims.** Set of user defined claims.
 - **Private claims.** Set of custom claims created to share information between parties that agree on using them.
- **Signature.** Ensures the integrity of the JWT header and payload.

Therefore, a JWT looks like: `header.payload.signature`.

Registered Claims

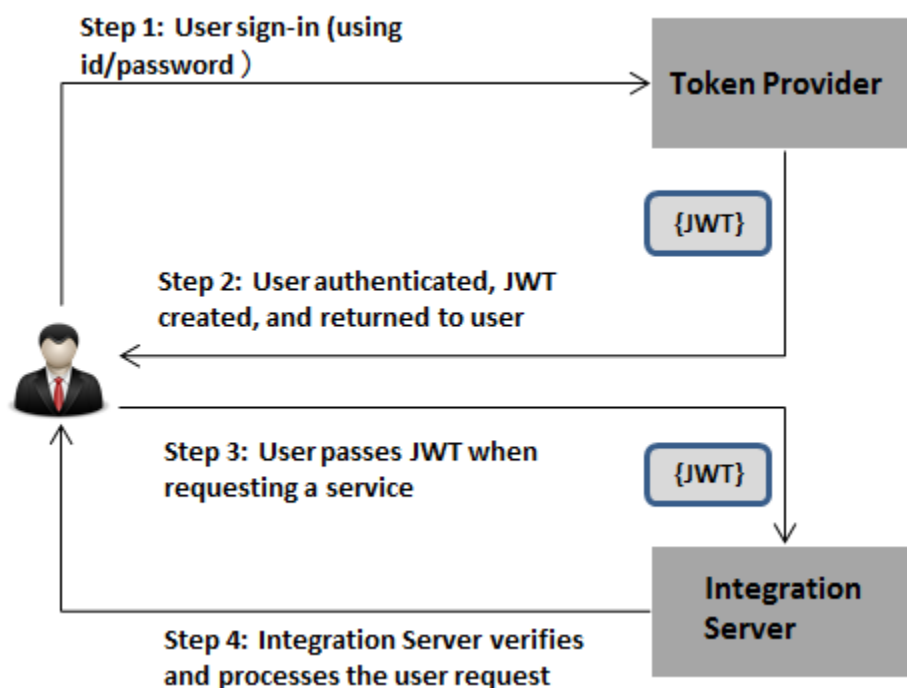
Registered claims are registered in the IANA JSON Web Token Claims registry. Following are some of the commonly used registered claims:

Claim Name	Description
iss	Issuer Identifies the principal that issued the JWT.
sub	Subject Identifies the principal that is the subject of the JWT.
aud	Audience Identifies the recipients that the JWT is intended for. Each principal intended to process the JWT must identify itself with a value in the audience claim.
exp	Expiration Time

Claim Name	Description
	Identifies the expiration time on or after which the JWT cannot be accepted for processing.
nbf	Not Before
	Identifies the time before which the JWT must not be accepted for processing.

Using JWT with Integration Server

You can use Integration Server to enable and configure JWT. The figure below illustrates the steps involved in JWT authentication.



- Step 1** User sign in (to Token Provider).
To access a protected data or a service residing on Integration Server, client sends a request for a JWT bearer token using his or her credentials to a third-party token provider.
- Step 2** Obtain JWT.
After authentication, the token provider creates a JWT and returns it to the client.
- Step 3** User uses the JWT.

Every time the client sends a request to access a resource or service in Integration Server, client sends a copy of this to token along with the request to Integration Server. Client sends the JWT in the authorization header using the bearer scheme to authenticate itself.

Step 4 Integration Server validates the JWT.

Based on the issuer to certificate mapping defined in Integration Server Administrator, Integration Server identifies the certificate alias and uses the corresponding public key to verify the signature. If signature verification fails, then the request is rejected.

Note:Integration Server does support unsigned JWTs (JWTs without a signature).

After the signature is verified, Integration Server verifies the claims contained in the JWT payload.

<u>For the claim name...</u>	<u>Integration Server verifies if...</u>
iss	The issuer id matches with the trusted issuer list defined in Integration Server.
sub	The user is mapped to an Integration Server.
aud	The audience value matches with the audience defined in the Global Claim Settings in Integration Server. This can be a list or a single value. If it is a list, then the defined Audience in Integration Server should be one of the value in this list.
exp	The "expiration time" is later than the current time. Integration Server factors in the Skew Mapping before rejecting a token.
nbf	The "not before time" is earlier to the current time. Integration Server factors in the Skew Mapping before rejecting a token.

If verification of any of above claim fails, then Integration Server rejects the request.

Step 5 Integration Server responds to the user.

After validation, Integration Server responds to the client with the requested data or with an appropriate error message.

Support for JWT in Integration Server

Currently, Integration Server supports only the following JWT features and functionality:

- Registered claim
- RSA Algorithms such as RS256, RS384, and RS512.

Configuring Integration Server to use JWT

Configuring Integration Server to use JWT involves the following stages:

- Stage 1** Add trusted issuers.
- Integration Server verifies the issuer id (`issclaim`) in the incoming JWT to check if it matches with the trusted issuers list that Integration Server maintains. For information about adding and editing trusted issuers, see [“Trusted Issuers” on page 407](#).
- Stage 2** Define a mapping between the issuer and certificate.
- During this stage, you define a mapping between the trusted issuer and the certificate. Based on the issuer to certificate mapping defined in this stage, Integration Server identifies the certificate alias and uses the public key from that certificate to verify the signature of the incoming JWT. For information about defining a mapping between issuer and certificate and deleting an existing mapping, see [“Issuer-Certificate Mapping” on page 408](#).
- Stage 3** Edit the Global Claim Settings.
- **Audience:** Integration Server verifies the audience defined in the incoming JWT to check if it matches with the list defined in the Global Claim Settings. Audience value can be a list or a single value. If it is a list, then the defined Audience in Integration Server should be one of the value in this list. For information about editing Global Claim Settings, see [“Editing Global Claim Settings” on page 411](#).
 - **Max Global Skew:** Enables you to define the permissible limits for a variation between any JWT issuer server clock and the Integration Server clock. Integration Server uses this value during verification of the incoming JWT. For more information about JWT clock skew settings, see [“Skew Mapping” on page 410](#).

Trusted Issuers

Trusted issuers are issuers whose public certificate is stored in Integration Server.

Integration Server maintains a list of trusted issuers. Using Integration Server Administrator you can add trusted issuers. You can also edit or delete an existing trusted issuer.

Adding Trusted Issuers

Complete the following steps to add a trusted issuer.

> To add a trusted issuer

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > JWT**.
3. Click **Trusted Issuers**.
4. Click **Add Issuer**.
5. On the Add Issuer screen, enter the issuer name in the **Name** field.

Note:

The issuer name is case-sensitive.

6. In the **Description** field, provide a description for the issuer.
7. Click **Save Changes**.

Once you add a trusted issuer, you can only edit the description for that issuer.

Deleting a Trusted Issuer

Complete the following steps to delete a trusted issuer.

Note:

You cannot delete an issuer for whom there exists a mapping (issuer-certificate mapping).

> To delete a trusted issuer

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > JWT**.
3. Click **Trusted Issuers**, and click the issuer that you want to delete.
4. Under Trusted Issuers, for the issuer that you want to delete, click **×** in the **Delete** column.
5. Click **Yes** to confirm.

Integration Server deletes the trusted issuer.

Issuer-Certificate Mapping

Using Integration Server Administrator, you can create a mapping between the trusted issuer listed in Integration Server and a certificate in a truststore. You can also delete an existing mapping.

Creating an Issuer-Certificate Mapping

An issuer-certificate mapping establishes an association between a trusted issuer and a truststore certificate. Integration Server uses the issuer-certificate mapping during verification of the signature of the incoming JWT.

Before creating an issuer-certificate mapping, ensure that the required trusted issuer, truststore alias, and certificate alias exist in Integration Server.

➤ To create an issuer-certificate mapping

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > JWT**.
3. Click **Issuer Configuration > Add Issuer Certificate Mapping**.
4. On the **Add Issuer Certificate Mapping** page, provide the following information:

In this field	Select
Issuer	The name of the trusted issuer that you want to map to a certificate.
Truststore Alias	The truststore alias for the truststore that contains the certificates of the signing authorities associated with the selected issuer.
Certificate Alias	The certificate alias for the certificate associated with the truststore alias.

5. Click **Save Changes**.

Deleting an Issuer Certificate Mapping

You can delete an issuer-certificate mapping when you want to remove some of the invalid or expired certificates, or certificates that you no longer need that are involved in any of the mappings.

➤ To delete an issuer-certificate mapping

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > JWT**.
3. Click **Issuer Configuration**.

4. In the **Issuer Certificate Mappings** table for the mapping that you want to delete, click **×** in the **Delete** column.

5. Click **Yes** to confirm.

Integration Server deletes the issuer-certificate mapping.

6. Click the **×** icon in the **Delete** column.

Skew Mapping

Sometimes the clocks on different servers may differ by a few seconds. As JWT uses timestamps, a variation between the JWT issuer server clock and the Integration Server clock can cause valid JWT tokens to be rejected. A Skew Mapping enables you to accommodate such clock differences up to the specified limit. Integration Server uses the Skew Mapping setting during verification of the incoming JWT.

Skew Mapping can be defined at the global level (as described in [“Editing Global Claim Settings” on page 411](#)) or at the issuer level (as described in [“Creating an Issuer Skew Mapping” on page 410](#)). The Issuer Skew Mapping value takes precedence over the Global Skew Mapping value if both values are available.

Creating an Issuer Skew Mapping

Before creating an issuer skew mapping, ensure that the required trusted issuer exists in Integration Server.

» To create an issuer skew mapping

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > JWT**.
3. Click **Issuer Configuration > Add/Update Issuer Skew Mapping**.
4. On the **Add/Update Issuer Skew Mapping** page, provide the following information:

In this field	Select
Issuer	The name of the trusted issuer.
Skew (seconds)	The maximum number of seconds for which the JWT issuer and Integration Server clocks can differ and still allow JWT expiry (exp) and not before (nbf) validations to succeed. This value takes precedence over the Max Global Skew value.

5. Click **Save Changes**.

Editing Global Claim Settings

Global Claim Settings maintain the properties for JWT that Integration Server looks for when verifying claims in an incoming JWT.

Complete the following steps to define the Global Claim Settings.

➤ To edit global claim settings

1. Open Integration Server Administrator.
 2. In the Navigation panel, select **Security > JWT**.
 3. Click **Edit Global Claim Settings**.
 4. On the Edit page, provide the following details:
 - **Audience:** A valid string or URI.
 - **Max Global Skew (Seconds):** Specifies the maximum number of seconds for which the JWT issuer and Integration Server clocks can differ and still allow JWT expiry (exp) and not before (nbf) validations to succeed. Provide a positive integer or zero.
- Note:**
This value is used if an Issuer Skew Mapping is not available for an issuer.
5. Click **Save Changes**.

19 Configuring Integration Server to Use Kerberos

■ Overview of Kerberos Usage in Integration Server	414
■ About Kerberos	414
■ Kerberos Delegated Authentication	415
■ Prerequisites to Configuring Kerberos	417
■ Limitations When Using Kerberos Authentication in Integration Server	417
■ Configuring Integration Server to Use Kerberos	417
■ JAAS Contexts for Kerberos	420
■ Troubleshooting Kerberos Configuration	421

Overview of Kerberos Usage in Integration Server

Kerberos is an authentication protocol that uses symmetric encryption and a trusted third party system to validate the identity of clients. The Kerberos protocol provides authentication over open and insecure networks in which communication between the hosts can be intercepted.

You can use Integration Server to enable and configure Kerberos authentication for service requests. Integration Server provides support for using Kerberos authentication with the following types of requests:

- Inbound and outbound HTTP and HTTPs requests at the transport level.
- Inbound and outbound HTTPS web service requests.

Note:

Integration Server currently supports Kerberos authentication for outbound web service requests of transportType *HTTPS* only.

About Kerberos

Kerberos authentication system consists of the following:

- A Kerberos client that needs to access and use Kerberos services.
- A trusted third-party system, specifically a key distribution center (KDC).
- A server that hosts services that are accessible using Kerberos authentication.

Kerberos authentication consists of the following phases:

1. **Authentication phase.** Client authenticates itself to the authentication service and requests a long-term ticket granting ticket (TGT).
2. **Service authorization phase.** Client uses the TGT to request a ticket for the specific service it wants to invoke.
3. **Service invocation phase.** Client sends the request to invoke the target service, including the service ticket obtained in the service authorization phase. If the server hosting the requested service authenticates the service ticket, the server invokes the requested service.

Kerberos Terminology

Before configuring Integration Server to use Kerberos authentication, you may find it helpful to first understand the following terminology:

- **Key Distribution Center (KDC).** The trusted third party system that provides authentication and ticket granting services and hosts the principal database. That is, the KDC consists of an authorization server, a ticket-granting server, and a database that contains the principals and their keys.

- **Realm.** All the computers that are managed by the KDC and secondary KDCs, if any, constitute the realm. That is, the realm includes all the nodes that share the same Kerberos database.
- **Principal.** A service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name in the following form:
principal-name.instance-name@realm-name
- **Service principal name.** The principal name of the service as registered with the principal database.
- **Keytab file.** The file consisting of a set of principals and their passwords. It can contain the principal password encrypted using different encryption algorithms.
- **Kerberos configuration file.** The file that contains configuration information such as Kerberos realm, location of KDC, defaults for the current realm, and encryption algorithms. Generally, this file is named "krb5.conf".
- **Subject.** The user or service that is authenticated by the JAAS login context.

Kerberos Delegated Authentication

Kerberos delegation allows Integration Server to reuse the client credentials to invoke another service which is hosted either on the same or different server. In this case, a principal (delegated user) can invoke a service on behalf of another principal (original requester).

Kerberos delegated authentication comprises the following phases:

1. **Authentication phase.** Client authenticates itself to the authentication service and requests a forwardable TGT (delegable token).
2. **Service authorization phase.**
 - a. Using the forwardable TGT the client requests for a service ticket for an intermediary.
 - b. The intermediary uses the forwardable TGT to request a service ticket for a service on behalf of the original requester.

Note:

Ticket is forwardable to any number of intermediaries. Irrespective of the number of intermediaries, the service request is invoked on behalf of the original requester.

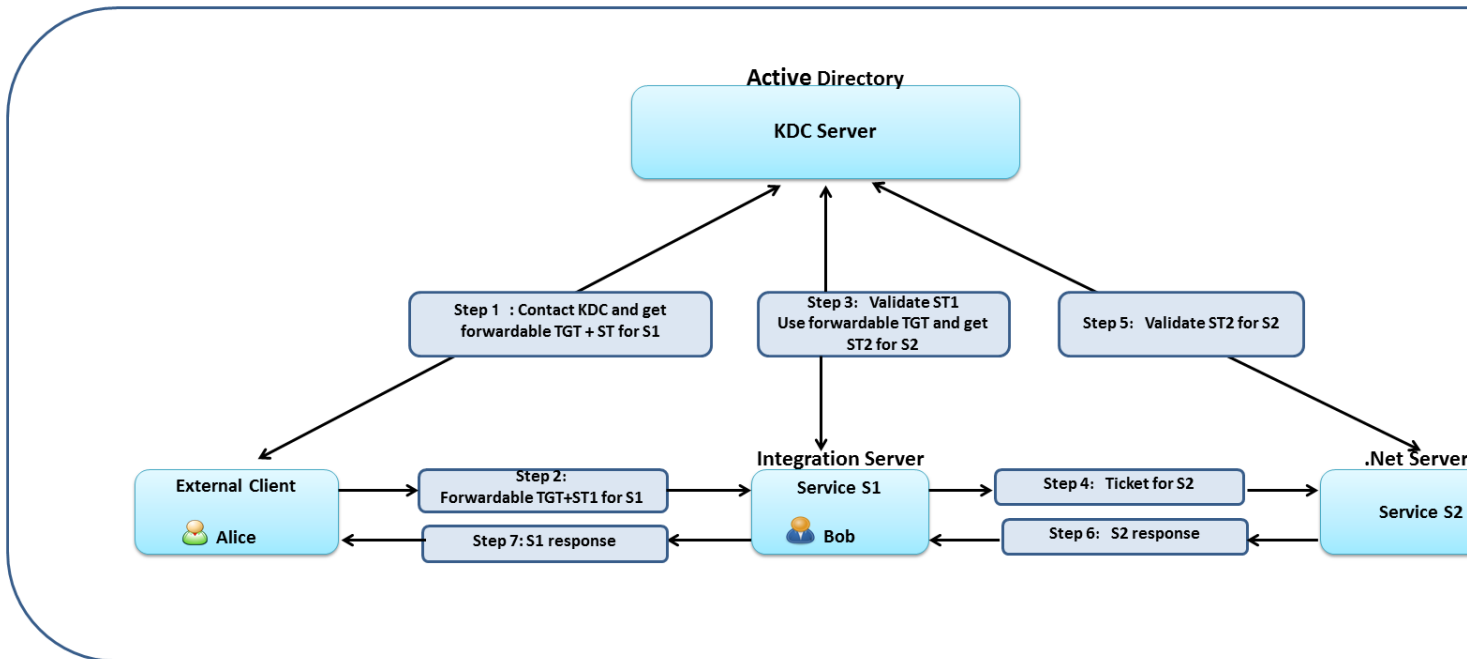
3. **Service invocation phase.** The intermediary sends the request to invoke the target service, including the service ticket obtained in the service authorization phase (step 2.b).

Example Use Case

Following is an example use case that describes the steps involved in Kerberos delegated authentication.

In this use case, consider the following:

- External client, intermediary (Integration Server), and the destination server (.Net Server) share the same KDC.
- Alice is a user account which the external client uses to access the service S1 in the intermediary (Integration Server).
- Bob is a user account that the intermediary uses to invoke the endpoint service S2 hosted on the .Net server.
- Alice invokes S1.



Settings:

- In S1, the input parameter `delegation` is set to `kerberos` in http outbound call (`pub.client:http`).

Steps:

1. External client contacts KDC and requests for a forwardable TGT and a service ticket (ST1) for Alice.
2. External client invokes S1 using the forwardable TGT and ST1.
3. Integration Server receives the token and then validates and extracts the forwardable TGT. Integration Server logs in as Bob and using the forwardable TGT of Alice requests a service ticket (ST2) for S2.
4. Integration Server invokes the target service S2 using ST2.
5. .Net Server contacts KDC and validates ST2. The service then gets invoked.
6. Integration Server receives the response for S2.
7. Integration Server forwards the response (success or error) to the external client.

Prerequisites to Configuring Kerberos

Before you configure Integration Server to use Kerberos authentication, you must ensure that:

- A working key distribution center (KDC) is set up.
- The KDC is configured as an LDAP directory, for authenticating incoming requests with Kerberos tickets.
- The KDC is configured as a user directory in Central Users or LDAP so Integration Server can identify and authorize the user that is part of the Kerberos ticket submitted by the client. For more information about Kerberos authentication, see [“Kerberos Authentication” on page 517](#).
- The Kerberos client is registered with the principal database of the KDC.
- The service that you want to access is registered with the KDC.
- A valid Kerberos configuration file is available.

Limitations When Using Kerberos Authentication in Integration Server

- Kerberos authentication in Integration Server currently supports only Kerberos login module that is provided by `com.sun.security.auth.module.Krb5LoginModule`.
- Integration Server might overwrite system properties such as `javax.security.auth.useSubjectCredsOnly` if they are already present in the client systems.
- The keytab configuration is certified on JDK 1.7_45 version and above.
- Kerberos Delegated Authentication is currently not supported for web service requests.

Configuring Integration Server to Use Kerberos

Use Integration Server Administrator to enable and configure Integration Server to use Kerberos authentication for inbound and outbound service requests. Integration Server passes this information to the Kerberos login module.

Keep the following information in mind when configuring Kerberos for Integration Server:

- Values specified for **Realm** and **Key Distribution Center Host** overwrite the default key distribution center (KDC) and realm set in the KDC configuration file specified in the **Kerberos Configuration File**. Do not specify values for **Realm** and **Key Distribution Center Host** if you do not want to overwrite the default KDC and realm in the Kerberos configuration file.
- If you specify **Key Distribution Center Host**, you must also specify **Realm**, and vice versa.
- By default, for outbound requests that require Kerberos authentication, Integration Server generates a Java Kerberos ticket using the JGSS Kerberos OID. If you need Integration Server to generate a SPNEGO-based Kerberos ticket for outbound requests that use Kerberos authentication, set the `watt.security.kerberos.client.useSPNEGO` server configuration parameter

to true. This instructs Integration Server to generate a SPNEGO token using SPNEGO OID (Object Identifier) for all outbound requests that require Kerberos authentication.

➤ **To configure Kerberos authentication**

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > Kerberos**.
3. Click **Edit Kerberos Settings**.
4. On the **Security > Kerberos > Edit** page, under Kerberos Settings, provide the following information:

In this field...	Specify...
Realm	<p>The domain name of the Kerberos server, in all uppercase letters. All the computers managed by the KDC and secondary KDCs, if any, constitute the realm.</p> <p>Example: <code>KERBEROS.RNDLAB.LOC</code></p> <p>This field is optional.</p> <div style="background-color: #f0f0f0; padding: 5px;"><p>Note: A value specified for Realm overwrites the realm set in the KDC configuration file specified in Kerberos Configuration File.</p></div>
Key Distribution Center Host	<p>The host name of the machine on which the KDC resides.</p> <p>Example: <code>lab.kerberos.rndlab.loc</code></p> <p>This field is optional.</p> <div style="background-color: #f0f0f0; padding: 5px;"><p>Note: A value specified for Key Distribution Center Host overwrites the default key distribution center set in the KDC configuration file specified in Kerberos Configuration File.</p></div>
Kerberos Configuration File	<p>The location of the Kerberos configuration file that contains the Kerberos configuration information, including the locations of KDCs, defaults for the realm and for Kerberos applications, and the host names and Kerberos realms mappings.</p>
Use Subject Credentials Only	<p>Specifies whether Integration Server requires a Kerberos V5 Generic Security Services (GSS) mechanism to obtain the</p>

In this field...**Specify...**

necessary credentials from an existing subject set up by the JAAS authentication module. Here, “subject” represents the user or service being authenticated in the JAAS login context.

When the **Use Subject Credentials Only** check box is selected, Integration Server requires a GSS mechanism to obtain the credentials from an existing Subject. Integration Server uses the Ticket Granting Ticket (TGT) stored in the subject to establish a GSS security context. The service ticket is also stored in the subject. When the **Use Subject Credentials Only** check box is selected, the JVM in which Integration Server runs can use only the credentials found in the Subject in the JAAS authentication module. The JVM cannot use another underlying mechanism to obtain the credentials.

When the Use Subject Credentials Only check box is cleared, Integration Server does not require a GSS mechanism to obtain credentials from an existing Subject. Instead, the JVM in which Integration Server runs can use another underlying mechanism of its choice, such as a reading from a protected file on disk, to obtain credentials for the Subject. The JVM first checks the Subject in the JAAS authentication module. If the JVM does not find the credentials in the JAAS Subject, then the JVM uses an alternate credential mechanism to obtain credentials.

The **Use Subject Credentials Only** check box must be selected if you want to use Kerberos authentication for service requests.

5. Click **Save Changes**.

Order of Precedence for Principal Name and Password

The Kerberos login module uses the principal name and password to authenticate the principal to the key distribution center (KDC). However, you can specify principal name and password in other locations.

- You can specify the principal name in the `is_jaas.cnf` file, which is the JAAS login configuration file, and the principal password in the `keytab` file. You must set `principal=client_principal_name` and `useKeyTab=true` in the `is_jaas.cnf` file to use the specific principal name and the corresponding password specified in the `keytab` file.

Note:

If you use this mode to specify the principal, the use of the Kerberos login module is restricted to that principal.

- For inbound service requests, you can also specify the principal name and password in the port configuration.
- For outbound services requests, you can specify the principal name and password in the `pub.client:http` service in the `clientPrincipal` and `clientPassword` fields in the `auth\kerberos` document.
- For inbound and outbound web service requests, you can also specify the principal name and password in the web service endpoint alias.
- For outbound web service requests, you can also specify the principal name and password at run time in the web service connector using the `clientPrincipal` and `clientPassword` in the `auth\message\kerberosSettings` document.

For service requests, Integration Server uses this order of precedence when determining which principal name and password to use:

1. The principal name in the `is_jaas.cnf` file and the corresponding password specified in the keytab file.
2. For inbound service requests, the principal name and password specified in the port configuration, if present.

For outbound service requests, the principal name and password specified in the `pub.client:http` service, if present.

For web service requests, Integration Server uses this order of precedence when determining which principal name and password to use:

1. The principal name in the `is_jaas.cnf` file and the corresponding password specified in the keytab file.
2. For outbound web service requests, the principal name and password specified at run time in the web service connector, if present.
3. For inbound and outbound web service requests, the principal name and password specified in the web service endpoint alias, if present.

JAAS Contexts for Kerberos

The `is_jaas.cnf` file distributed with Integration Server is located in the `Integration Server_directory\instances\instance_name\config` folder and includes the following JAAS contexts:

- `IS_KERBEROS_INBOUND` for use with inbound requests.
- `IS_KERBEROS_OUTBOUND` for use with outbound requests.

You can use the JAAS contexts supplied in `is_jaas.cnf` or add your own.

Important:

However, the default JAAS login contexts—`IS_KERBEROS_INBOUND`, `IS_KERBEROS_OUTBOUND`, `IS_Transport`, `PlatformManagement`, `WSS_Message_IS`, `WSS_Transport_IS`, and `WSS_Transport_ProxyService`—should not be deleted from the JAAS configuration file.

Troubleshooting Kerberos Configuration

I receive the following error when I execute the web service connector: org.apache.axis2.AxisFault: Error in building kerberos token.

Possible reasons for this error are:

- The format of the JAAS context is incorrect.
- There are no Kerberos login modules configured for the specified principal name.
- The absolute path of the keytab file specified in the JAAS context is incorrect.
- The encryption algorithm in the keytab file is different from the one used by KDC.
- The JDK version you are using is lower than JDK 1.7_45. The keytab configuration is certified on JDK 1.7_45 version and above.

I receive the following error when I execute the web service connector: Pre-authentication information was invalid (24) KrbException: Identifier doesn't match expected value (906).

Possible causes for this error are:

- The truststore or the keytab file is corrupt. In such cases, Software AG recommends that you recreate these files and place them in a location that is not frequently accessed.
- Your system time and the time in the system that hosts the KDC are not the same. The clocks in these two systems cannot be out of sync for by more than 300 seconds, which is the default clockskew.

I receive the following error when I request for Kerberos delegation by setting delegation to kerberos: Kerberos delegatable credentials do not exist.

Possible reasons for this error is: The *requestDelegatableToken* parameter in *pub.client:http* service is not set to *true*.

20 Setting Up HTTP URL Aliases

■ Overview	424
■ Creating an HTTP URL Alias	425
■ Enabling Partial Matching of URL Aliases	430
■ Displaying HTTP URL Aliases	431
■ Editing a URL Alias	432
■ Deleting a URL Alias	432
■ Portability of URL Aliases and Possible Conflicts	433

Overview

An HTTP URL alias is an alias that you create to replace a portion of the URL to a resource on Integration Server. The URL alias typically replaces the path portion of the URL which identifies the name and location of the resource. For example, when you create a URL alias for a service, the alias replaces the invoke directive and the name and location of the service on Integration Server. A client request for the service includes the alias instead of the invoke directive and the service information. When Integration Server receives the request, Integration Server invokes the service associated with the alias.

You can associate a URL alias with items such as services, HTML pages, dynamic server pages (DSP), an image file, or web services. In addition to associating a URL alias with a single resource, you can also identify a different resource for a URL alias based on the incoming port for the request. This makes it possible to define a single URL alias that resolves to different destinations based on the incoming port of the HTTP request. In addition, you can also define the "empty path" as an alias, allowing the definition of a default destination for any incoming port.

URL aliases have several benefits:

- A URL alias may be easier to type than full URL path names.
- If a resource has a URL alias, it can be easier to move resources on Integration Server. An administrator updates the alias information to point to the new location; there is no need to notify users of the change.
- A URL alias is more secure than URL path names because users do not see directory, service, or file names.

Note:

A URL alias can be used with HTTP or HTTPS ports only.

You can create a URL alias from Designer or Integration Server Administrator.

Create an alias from Integration Server Administrator if you want to:

- Assign an alias to a resource other than a Flow, Java, C, Adapter, or XSLT service. For example, from Integration Server Administrator, you can associate an alias with a DSP, HTML, jpg, or web service.
- Create a URL alias for a REST resource configured using the legacy approach.

Note:

For more information about the approaches for configuring REST resources, see *REST Developer's Guide*.

- Associate more than one URL alias with a particular resource.
- Create an alias for the "empty path".
- Specify a port-specific destination for a particular URL alias.

If you want to create a single URL alias for a service, consider using Designer. When you use Designer to create a URL alias, you do not need to specify the URL path. If Designer was used to create the URL alias, you can move a service from one folder to another without having to update the path name assigned to the alias. For more information about creating URL aliases for services in Designer, see *webMethods Service Development Help*.

Creating an HTTP URL Alias

When you create a URL alias, you create an association between an alias and a resource on Integration Server. Keep the following information in mind when creating a URL alias:

- You can create a URL alias for a request that comes through an HTTP or HTTPS port only.
- An alias name must be unique across Integration Server.
- You can associate a single URL alias with multiple destinations by specifying port mappings. A port mapping correlates the alias with a different URL alias based on the port on which the request was received. For more information about creating port mappings for a URL alias, see [“Using Port Mappings with a URL Alias” on page 427](#).
- You can create a URL alias for the empty path (/) for a hostname:port combination, such as localhost:5555. By creating an empty path alias, you can specify a default destination for any incoming port. For more information about the empty path alias, see [“Using a URL Alias for the “Empty Path”” on page 429](#).
- If you want to use URL alias with a REST resource you must enable partial matching of URL aliases. For more information, see [“Enabling Partial Matching of URL Aliases” on page 430](#).
- If you enabled or intend to enable partial matching of URL aliases, do not define an alias that begins with another alias. For more information, see [“Enabling Partial Matching of URL Aliases” on page 430](#).
- You can create an HTTP URL alias for a service using Designer as well as Integration Server. However, if you want to use the same URL alias for more than one service or a service and other resources, you must create the URL alias using Integration Server Administrator. For instructions on creating an HTTP URL alias from Designer, see *webMethods Service Development Help*.
- As of 10.3, Microservices Runtime or Integration Server equipped with a Microservices Runtime license creates two predefined URL aliases named “health” and “metrics” for use with the `health` and `metrics` endpoints that provide monitoring capabilities. Software AG recommends that you avoid creating URL aliases with these names. Additionally, Software AG does not recommend editing the predefined “health” or “metrics” URL aliases.

Note:

If you migrate to Microservices Runtime version 10.3 or higher from an earlier version and you already have a URL alias named “health” and/or “metrics”, Microservices Runtime does not create the “health” and/or “metrics” URL alias. Any invocation of the `health` or `metrics` endpoints does not result in execution of health indicators or metrics gathering, respectively. If you want to use the health gauge and/or metrics gathering, you need to rename your existing URL alias named “health” or “metrics”. Upon restart, Microservices

Runtime will create a new health and/or metrics URL alias that corresponds to the health endpoint and/or metrics functionality.

➤ To create an HTTP URL alias

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.
3. Click **Create URL Alias**.
4. Under **URL Alias Properties**, provide the following information:

For this parameter...	Specify...
Alias	<p>A name for the alias.</p> <p>The alias name cannot include a space, nor can it include the following characters:</p> <p># % ? ' " < \</p> <p>The alias name cannot begin with the string "http://"</p> <p>There is no length restriction for the alias.</p> <p>Note: Do not specify an alias if you intend to use this URL alias for the empty path.</p>
Use as empty path alias	<p>Select this check box if you want to use this alias for the empty path.</p> <p>When you select the Use as empty path alias check box, Integration Server changes the alias to <EMPTY>.</p>
Default URL Path	<p>The path to the resource on Integration Server. For details on how to specify a URL path for the different types of destinations, see “Specifying the URL Path” on page 427.</p> <p>You must specify a Default URL Path field if you do not define any port mappings for the URL alias.</p> <p>If the URL alias includes port mappings, the Default URL Path field is optional. However, it might be useful to supply a Default URL Path value so that a request received on a port for which there is not a port mapping will “fall through” to the default URL path. If a URL alias specifies a port mapping but does not specify a Default URL Path value, Integration Server uses the alias as the URL path.</p>

For this parameter...	Specify...
	The URL path cannot include a space or the following characters: # % ? ' " < \
Package	Name of the package with which you want to associate this alias.

- If you want to define a different destination for the URL alias based on the incoming port, create a port mapping. For information, see [“Using Port Mappings with a URL Alias” on page 427](#). do the following:
- Click **Save Changes**.

Specifying the URL Path

The URL path defines the resource to which the alias resolves. You specify a URL path for the URL alias and/or for port mappings defined for a URL alias.

Following are examples of formats for the URL path for different resources:

- If the resource is a service:
invoke/folder_name.subfolder_name/service_name
- If the resource is a REST resource:
rest/folder_name/subfolder_name
- If the resource is a REST V2 resource:
restv2/folder_name:resource_name
- If the resource is not a service or a REST resource:
package_name/file_name.ext

Note:

The URL path cannot include a space or the following characters:

% ? ' " < \

Using Port Mappings with a URL Alias

For a URL alias, you can create one or more port mappings which associates a port with a resource. Because the URL path can be different for each port, using port mappings allows the use of a single URL alias with multiple destinations. You might want to define a different destination for each business partner.

When Integration Server receives a request that contains a URL alias, Integration Server resolves the request destination by first determining if there is a URL path associated with the incoming

port. If there is not a URL path mapped to the port number, then Integration Server uses the default URL path for the alias as the request destination. Port mappings are always evaluated first.

For example, in the following URL alias named “Order”, when an HTTP request that includes the alias “Order” is received on the port 6677, Integration Server invokes the service `processing.expedite.processOrder`. For a HTTP request that uses the alias “Order” that is received on any port besides 6677, Integration Server invokes the service `Order:processsOrder`.

URL Alias Properties		
Alias	<input type="text" value="Order"/>	<input type="checkbox"/> Use as empty path alias
Default URL Path	<input type="text" value="invoke/Order/processOrder"/>	
Package	<input type="text" value="Processing"/>	
Association	Package	
Port Mappings		
Port Number	URL Path	Delete Mapping
6677	<code>invoke/processing.expedite/processOrder</code>	
Add Port Mapping		
Port Number	URL Path	Add Mapping
<input type="text"/>	<input type="text"/>	

Adding a Port Mapping to a URL Alias

You can create multiple port mappings for a URL alias. Keep the following points in mind when defining a port mapping for a URL alias:

- You can define a port mapping for a URL alias created in Integration Server Administrator only. You cannot define a port mapping for URL alias defined for a service using Designer.
- Incoming requests with the URL alias received on a port for which there is not a port mapping resolve to the path specified in the **Default URL Path** field. If the alias does not specify a **Default URL Path** value, Integration Server uses the alias as the URL path.

➤ To add a port mapping to a URL alias

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.
3. In the HTTP URL Alias list, select the URL alias for which you want to define port mappings.
4. Under Port Mappings, in the **New Port Number** list, select the port for which you want to specify an alternate destination. Integration Server lists all of the defined HTTP and HTTPS ports.
5. In the **New URL Path** field, specify the path to resource that you want to use as the destination. The URL path cannot include a space or the following characters: # % ? ' " < \

For information about how to specify different resources, see [“Specifying the URL Path” on page 427](#).

6. Under **Add Mapping**, click **+** to add the port mapping to the alias.
7. Repeat steps 4 to 6 for each port for which you want to specify a different destination than that in the **Default URL Path** field.
8. Click **Save Changes**.

Deleting a Port Mapping for a URL Alias

You can delete any of the port mappings added to a URL alias.

Note:

If you intend to delete all of the port mappings for a URL alias, make sure the URL alias specifies a **Default URL Path** value.

➤ To delete a port mapping for a URL alias

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.
3. In the HTTP URL Alias list, select the URL alias for which you want to delete a port mapping.
4. Under Port Mappings, click **×** in the **Delete** column for the port mapping that you want to delete.
5. Repeat the preceding step for each port mapping that you want to delete for an alias.
6. Click **Save Changes**.

Using a URL Alias for the “Empty Path”

You can create a URL alias for the “empty path” for any HTTP/S port. You can define one destination to be used by all HTTP/S ports. Or, by adding a port mapping for each port, you can define a different default destination for the empty path for any incoming HTTP/S port. You can also define port mappings for the empty path for particular ports and a default destination for all other parts.

Integration Server considers either of the following to be an empty path:

- *host:port*
- *host:port/*

A URL alias for the empty path has a predefined alias name of <EMPTY>. Each Integration Server can have one <EMPTY> alias only.

Note: Integration Server Administrator can be accessed with the URL request *host:port/WmRoot*.

Creating a URL Alias for the Empty Path

You can create a new URL alias for use as the empty path alias or you can convert an existing alias created in Integration Server Administrator to the empty path alias. Note that a URL alias created for a service using Designer cannot be used as the empty path alias.

> To create a URL alias for the empty path

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.
3. Do one of the following:
 - In the HTTP URL Alias list, select the URL alias that you want to use as the empty path alias. On the **Settings > URL Alias > aliasName > Edit** page, select the **Use as empty path alias** check box. Integration Server changes the alias name to <EMPTY>.
 - Click **Create URL Alias**. On the **Settings > URL Alias > Create** page, select the **Use as empty path alias** check box. Integration Server changes the alias to <EMPTY>.

Specify the remaining configuration information for a URL alias. For more information about creating a URL alias, see [“Creating an HTTP URL Alias” on page 425](#).
4. Click **Save Changes**.

Enabling Partial Matching of URL Aliases

In some cases, URL requests include identifiers for a particular resource. Because these identifiers vary for each instance of a resource, URL requests might not exactly match any of the defined URL aliases for a particular resource. To enable you to define URL aliases for such resources, Integration Server can use partial matching to process URL requests. A *partial match* occurs when a request URL matches or begins with only part of a URL alias.

When partial matching is enabled and Integration Server receives a request URL, an alias is considered a match if the entire alias matches all or the beginning of the request URL, starting with the first character of the request URL's path.

For example, if URL alias a2 has a URL path of `rest/purchasing/invoice` and Integration Server receives the following request URL:

```
http://MyHost:5555/a2/75909
```

The request URL matches the a2 alias and resolves to the path:
`http://MyHost:5555/rest/purchasing/invoice/75909`.

Integration Server retains the trailing characters of the request URL. In this case, Integration Server retains the `/75909`.

Note:

If you want to use URL alias with a REST resource configured using the legacy approach, you must enable partial matching of URL aliases. For information about defining URL aliases for REST resources, see *REST Developer's Guide*.

> To enable Integration Server to use partial matching of URL requests

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Extended**.
3. Click **Edit Extended Settings**.
4. Add the following to the **Extended Settings** box:
`watt.server.url.alias.partialMatching=true`.
5. Click **Save Changes**.
6. Restart Integration Server.

Displaying HTTP URL Aliases

To display a list of *all* HTTP URL aliases defined on Integration Server, you must view them from Integration Server Administrator.

> To display a list of all URL aliases

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.

About HTTP URL Alias Association

The **Settings > URL Aliases** page displays aliases that were created from Integration Server Administrator and aliases that were created from Designer. The contents of the **Association** column indicate the tool used to create the URL alias.

Value	Description
Package	Indicates the alias was created from Integration Server Administrator.
<i>service_name</i>	Indicates the alias was created from Designer. When you create an alias from Designer, you can view it from Designer or Integration Server. In addition, you can move a service from one folder to another without having to update the path name assigned to the alias.
Server	Indicates that the alias was created in an earlier version of Integration Server and was migrated to this Integration Server. The alias works at this point, but the only change you can make to it is to update the package association. Once you update the package association, you can view and change the alias as you would other HTTP URL aliases.

From Designer, you can display the HTTP URL alias assigned to an individual service. Designer displays a service's alias only if the alias was created from Designer or Developer.

Note:

When an HTTP URL alias is migrated from an earlier release of Integration Server, the alias is displayed on the HTTP URL Alias List screen with "Server" in the **Association** field and nothing in the **Package** field. The alias is functional at this point, but the only change you can make to it is to update the package association. Once you update the package association, you can view and change the alias as you would other HTTP URL aliases.

Editing a URL Alias

You can use Integration Server Administrator to edit URL aliases created in Integration Server Administrator only. You cannot edit URL aliases created in Designer.

➤ **To edit a URL alias**

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.
3. In the HTTP URL Alias list, select the URL alias that you want to edit.
4. Make the changes to the URL alias.
5. Click **Save Changes**.

Deleting a URL Alias

You can use Integration Server Administrator to delete URL aliases created in Integration Server Administrator or in Designer.

Note: Integration Server Administrator automatically deletes all URL aliases associated with a package when you delete or disable the package.

➤ To delete a URL alias

1. Open Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Settings > URL Aliases**.
3. Locate the row that contains the alias you want to delete, and click the **×** icon.

Integration Server prompts you to confirm that you want to delete the alias.

4. Click **OK**

Portability of URL Aliases and Possible Conflicts

From Designer, you can move a service from one folder to another and the alias will stay with the service.

From Integration Server, when you use the Integration Server package replication feature to copy a package from one Integration Server to another, any HTTP URL aliases associated with the package stay with it. This is true whether the alias was created from Integration Server or from Designer or Developer.

When you receive a package from another Integration Server, it is possible that an alias associated with the new package has the same name as an HTTP URL alias already defined on your Integration Server.

Integration Server maintains a registry of HTTP URL alias names, and displays the contents of this list on the **Settings > HTTP URL Alias** screen. Integration Server builds this list when it loads packages during server startup, and when an alias is added, edited, or deleted. In cases where two packages are associated with the same alias name, Integration Server uses the alias associated with the first of the two packages to load. When the second package is loaded, Integration Server flags its associated alias as a duplicate, and does not add it to the registry.

In other words, Integration Server uses package load order to determine precedence. WmRoot is always loaded first. The order in which other packages are loaded depends on other factors, such as package dependencies and operating system, and therefore cannot be easily predicted.

For this reason, when you install a package that was published by another Integration Server, check the details screen for the package (Packages > Management > *package_name*) for load warnings.

21 Using HTTP Interceptors

■ Overview of HTTP Interceptors	436
■ Considerations for Creating an HTTP Interceptor	437
■ Creating an HTTP Interceptor	438

Overview of HTTP Interceptors

Integration Server provides a framework for creating HTTP interceptors that intercept and process HTTP requests and responses. HTTP interceptors have read-only access to HTTP requests and responses. They can intercept all HTTP requests and responses, with the exception of: OAuth messages; services that use their own HTTP URL connections; and intermediate requests and responses in NTLM, DIGEST and REDIRECTS.

Types of HTTP Interceptors

There are two types of HTTP interceptors:

- Inbound interceptors, which are invoked when Integration Server processes inbound HTTP requests. Here, Integration Server is the HTTP server.
- Outbound interceptors, which are invoked when Integration Server sends outbound HTTP requests. Here, Integration Server is the HTTP client.

You can implement either or both types of interceptors to perform tasks, such as writing HTTP requests to a data lake or a log file.

HTTP Interceptor Interfaces

An HTTP interceptor can be created by writing a Java class that implements one of the following interfaces:

- `com.softwareag.is.interceptor.HttpInterceptorIFC` for inbound interceptors.
- `com.softwareag.is.interceptor.HttpOutboundInterceptorIFC` for outbound interceptors.

Both interfaces declare two methods that the implementing classes can implement for processing requests and responses:

- The `preProcess` method, which is invoked for HTTP requests. In the case of inbound interceptors it is invoked for the HTTP requests received by the Integration Server, and in the case of outbound interceptors for the HTTP requests sent by Integration Server.
- The `postProcess` method, which is invoked for HTTP responses. In the case of inbound interceptors it is invoked for the HTTP responses sent by the Integration Server. In the case of outbound interceptors for the HTTP responses received by Integration Server.

Working of HTTP Interceptors

Integration Server makes the raw HTTP requests and responses, including the HTTP headers, status codes, and status messages, accessible to HTTP interceptors.

The working of inbound interceptors is as follows:

- Requests: Integration Server first authenticates the client and then invokes the inbound HTTP interceptor. The `preProcess` method of the inbound interceptor has access to the raw HTTP

request data, including the headers, and a clone of the request body. The `preProcess` method is called before Integration Server does any processing, such as engaging a content handler on the request. If required, the `preProcess` method can stop the normal processing of a request by throwing the `HttpInterceptorException`. In the case of inbound interceptors, this exception contains the HTTP status code and the message that Integration Server sends as a response to the requesting client.

- Responses: The inbound HTTP interceptor intercepts the HTTP response just before Integration Server sends it to the client. The `postProcess` method of the inbound interceptor can access a clone of the response body, along with the response status code, status message, and headers, before Integration Server writes the response to the connection.

The working of outbound interceptors is as follows:

- Requests: Integration Server invokes the `preProcess` method of the outbound HTTP interceptor just before sending the request to the provider. The `preProcess` method has access to the raw HTTP request data, including the headers, and a clone of the request body. If required, the `preProcess` method can prevent Integration Server from sending the request by throwing an `HttpInterceptorException`.
- Responses: Integration Server clones the response it receives and invokes the `postProcess` method of the outbound interceptor. The `postProcess` method processes the cloned response in parallel to Integration Server response processing.

Considerations for Creating an HTTP Interceptor

You need to consider the following when creating an HTTP interceptor:

- An HTTP interceptor class must implement the `com.softwareag.is.interceptor.HttpInterceptorIFC` interface for inbound messages and the `com.softwareag.is.interceptor.HttpOutboundInterceptorIFC` for outbound messages.
- You can register only one HTTP interceptor class of each type.
- The HTTP interceptor class must provide implementations of the `preProcess` and `postProcess` methods. For more information, see the *webMethods Integration Server Java API Reference*.
- Using an HTTP interceptor delays the processing of a request. Any logic that the `preProcess` method executes to determine whether or not to continue with request processing needs to be done synchronously. However, if the `preProcess` method includes logic that can be done asynchronously by spawning another thread, regular processing can proceed without a delay.
- Using an HTTP interceptor also delays the processing of responses.
- An HTTP interceptor cannot modify the content of the HTTP request or response.
- Integration Server passes a byte array representation of the input stream in the case of an inbound interceptor (or output stream in the case of an outbound interceptor).

Note:

Ensure that you have sufficient system resources for the payloads that need be processed by the outbound interceptor.

- Unless an HTTP interceptor throws an `HttpInterceptorException` to reject a request, Integration Server continues with the normal processing of the original input (or output) stream.

Creating an HTTP Interceptor

You can create and add one or both interceptor classes using the procedure given below.

➤ To create and add an HTTP interceptor

1. Write a Java class that implements the required (`HttpInterceptorIFC` or `HttpOutboundInterceptorIFC`) interface.
2. Build the class and add it to a jar file.
3. Place the jar file in one of the following locations that Integration Server adds to the classpath:
 - `Software AG_directory \ Integration Server_directory \ instances \ instanceName \ lib \ jars \ custom`
 - `Software AG_directory \ Integration Server_directory \ lib \ jars \ custom`
 - `Software AG_directory \ Integration Server_directory \ instances \ instanceName \ packages \ packageName \ code \ jars \ static`
4. Set the following server configuration properties in Integration Server:
 - For an inbound interceptor:
 - `watt.server.http.interceptor.enabled`: Set this parameter to true to use inbound HTTP interceptors.
 - `watt.server.http.interceptor.impl`: Set this parameter to the fully qualified name of the class that implements the inbound interceptor interface.
 - `watt.server.http.interceptor.preprocess.sizeLimit`: Set this parameter to define a size limit on the requests that will be handled by the inbound interceptor. The parameter is not applicable to the outbound interceptor.
 - For an outbound interceptor:
 - `watt.server.http.interceptor.outbound.enabled`: Set this property to true to use outbound HTTP interceptors.
 - `watt.server.http.interceptor.outbound.impl`: Set this property to the fully qualified name of the class that implements the outbound interceptor interface.

For more information about these parameters, see [“watt.server.” on page 1053](#).

5. Restart Integration Server.

22 Configuring Integration Server for HTTP Compression

■ Overview	440
■ HTTP Request Compression	442
■ HTTP Response Compression	443

Overview

Integration Server supports data compression, also referred as content encoding, for HTTP requests and HTTP responses:

- When Integration Server is acting as an HTTP client and sending compressed payload in HTTP requests.
- When Integration Server is acting as an HTTP client and receiving compressed payload in HTTP responses.
- When Integration Server is acting as an HTTP server and receiving compressed payload in HTTP requests.
- When Integration Server is acting as an HTTP server and sending compressed payload in HTTP responses.

HTTP compression schemes improve the data transfer speed and make better network bandwidth utilization. If user has a large set of data and wants to send it via HTTP request, Integration Server supports to compress the data before sending it and also supports to compress the data before sending the HTTP response.

Integration Server supports the following HTTP compression schemes:

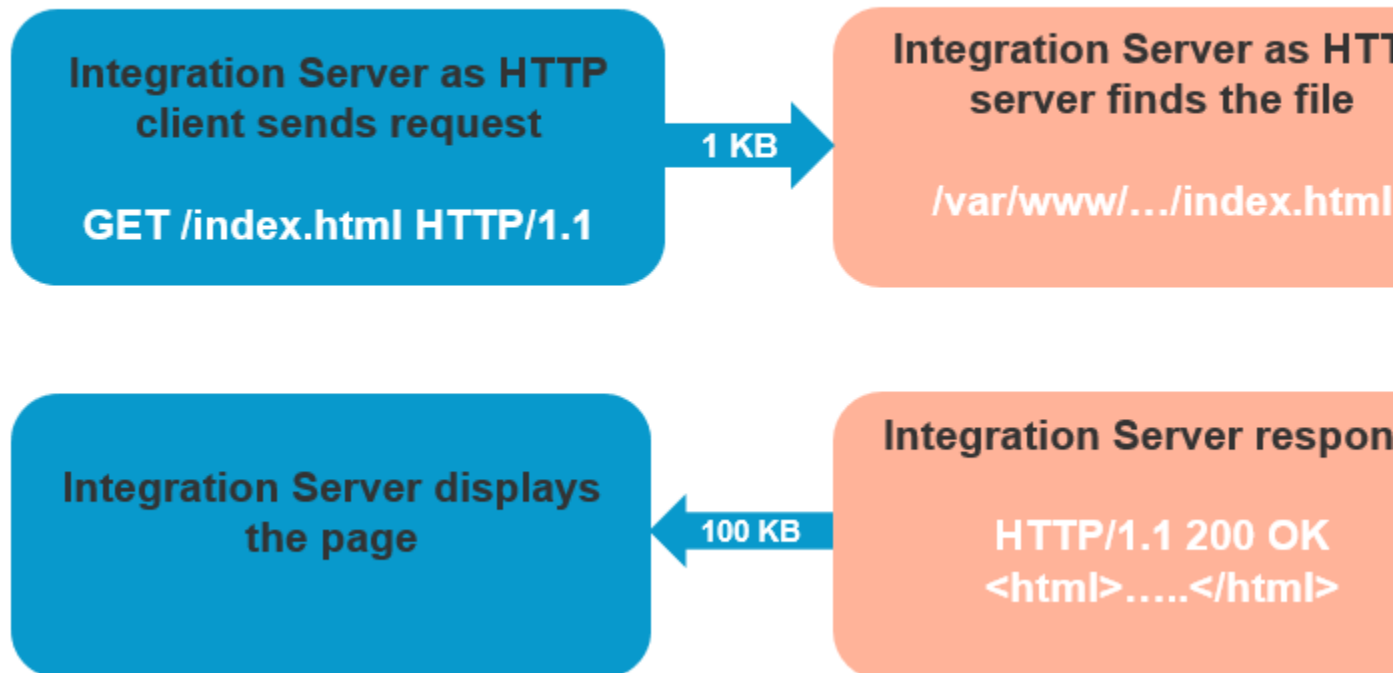
- Gzip
- Deflate

Note:

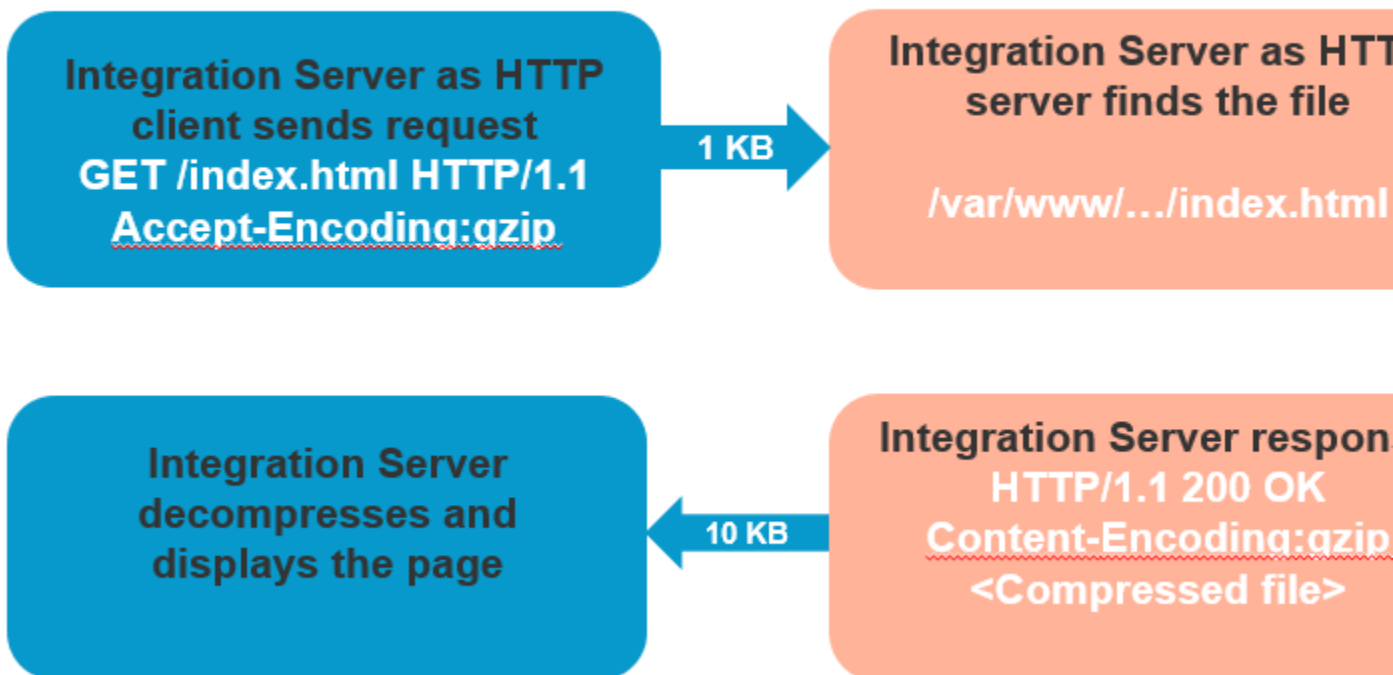
Integration Server does not support HTTP request or response compression for dynamic server pages (DSPs).

Integration Server as HTTP client supports data compression before sending the HTTP request and can also instruct the HTTP server to compress the data before responding back to the client.

Standard HTTP Request Response



Compressed HTTP Request Response



When Integration Server is acting as an HTTP client, you can use the built-in services:

- `pub.compress:compressData` to compress the data before sending the HTTP request.

- `pub.compress:decompressData` to decompress the data after receiving the HTTP response.

For more information about these services, see the *webMethods Integration Server Built-In Services Reference* guide.

When Integration Server is acting as an HTTP server, you must configure the following server configuration parameters as `true`:

- `watt.server.http.request.supportCompression` to support HTTP request compression.
- `watt.server.http.response.supportCompression` to support HTTP response compression.

For more information about this parameter, see [“watt.server.” on page 1053](#).

HTTP Request Compression

When Integration Server is acting as an HTTP client and if user has a large set of data to compress, then user can use the `pub.compress:compressData` service to compress the data. While executing the service, user can define any supported compression scheme to compress the data. When client sends the HTTP request, it must define the `Content-Encoding` HTTP header as `gzip` or `deflate`. This indicates which compression scheme the client has used to compress the request payload.

When Integration Server is acting as an HTTP server and supports the HTTP compression where the server configuration parameter, `watt.server.http.request.supportCompression` is set as `true`; after receiving the HTTP request, Integration Server uses the same compression scheme mentioned in `Content-Encoding` to decompress the data before sending the data for further processing.

If the HTTP request contains multiple algorithm schemes in `Content-Encoding`, the HTTP server uses the first valid compression scheme mentioned. Consider the following examples of HTTP headers containing multiple algorithm schemes.

```
Content-Encoding: deflate, gzip
```

In this case, Integration Server uses the `deflate` compression scheme.

```
Content-Encoding: <UnsupportedType>, gzip
```

In this case, Integration Server uses the `gzip` compression scheme.

```
Content-Encoding: deflate, <UnsupportedType>
```

In this case, Integration Server uses the `deflate` compression scheme.

```
Content-Encoding: <UnsupportedType>, <UnsupportedType>
```

In this case, Integration Server ignores the HTTP request.

Example of an HTTP request compression:

```
Request
POST http://localhost:<port>/restv2/testEncoding:ResourceForEncoding/testRequest
  Content-Encoding: gzip
Response
HTTP/1.1 200 OK
```

HTTP Response Compression

When Integration Server is working as an HTTP client and the client is expecting a large set of data as a response from the HTTP server, then client can request the server to compress the data before responding back to the client. When client sends the HTTP request with the `Accept-Encoding` HTTP header as `gzip` or `deflate`, this indicates which compression scheme the server can use to compress the response payload.

When Integration Server is acting as an HTTP server and supports the HTTP compression where the server configuration parameter, `watt.server.http.response.supportCompression` is set as `true`; while sending the HTTP response, Integration Server defines the `Content-Encoding` HTTP header as `gzip` or `deflate`. This indicates which compression scheme the server has used to compress the data.

Integration Server acting as an HTTP client receives the compressed response from the server and user can use the `pub.compress.decompressData` service to decompress the data. The service uses the same compression scheme mentioned in the HTTP header to decompress the data.

If the HTTP request contains multiple algorithm schemes in `Accept-Encoding`, the HTTP server uses the first valid compression scheme mentioned. Consider the following examples of HTTP headers containing multiple algorithm schemes.

```
Accept-Encoding: deflate, gzip
```

In this case, Integration Server uses the deflate compression scheme.

```
Accept-Encoding: <UnsupportedType>, gzip
```

In this case, Integration Server uses the gzip compression scheme.

```
Accept-Encoding: deflate, <UnsupportedType>
```

In this case, Integration Server uses the deflate compression scheme.

```
Accept-Encoding: <UnsupportedType>, <UnsupportedType>
```

In this case, Integration Server responds with the uncompressed data.

Examples of an HTTP response compression:

```
Request
GET http://localhost:<port>/rad/testEncoding:ResourceForEncoding/testResponse
  Accept-Encoding: gzip
Response
HTTP/1.1 200 OK
Content-Encoding: gzip

Request
GET http://localhost:<port>/rad/testEncoding:ResourceForEncoding/testResponse
  Accept-Encoding: deflate
Response
HTTP/1.1 200 OK
Content-Encoding: deflate
```

```
Request
GET http://localhost:<port>/invoke/folder:serviceName
Accept-Encoding: gzip, deflate
Response
HTTP/1.1 200 OK
Content-Encoding: gzip
```

```
Request
GET http://localhost:<port>/odata/folder:odataEntityName/Resource
Accept-Encoding: deflate, gzip
Response
HTTP/1.1 200 OK
Content-Encoding: deflate
```


23 Configuring Integration Server to Connect to an SFTP Server

- Overview of SFTP 446
- Creating an SFTP Server Alias 446
- Creating an SFTP User Alias 450

Overview of SFTP

The SSH File Transfer Protocol (SFTP) is a network protocol that is based on the Secure Shell protocol (SSH). SFTP facilitates secure file access, file transfer, and file management over any reliable data stream.

You can configure Integration Server to connect to an SFTP server to perform the following tasks using the SFTP protocol:

- Transfer files between Integration Server and the SFTP server. You can get a file from the SFTP server and store it in the local machine or upload a file from your local machine to the SFTP server.
- Access files in the SFTP server. You can view the directories and files in the SFTP server and also view their permissions and ownership information.
- Manage directories or files in the SFTP server. You can create, rename, or delete files or directories in the SFTP server. You can also change the permissions or ownership of files in the SFTP server.

You can use Integration Server Administrator to define the following SFTP aliases:

- **SFTP server alias.** The SFTP server alias contains configuration parameters that Integration Server uses to connect to an SFTP server.
- **SFTP user alias.** The SFTP user alias contains client configuration parameters that Integration Server uses to authenticate and function as an SFTP client.

Creating an SFTP Server Alias

An SFTP server alias is a named set of parameters that Integration Server uses to connect to an SFTP server.

Important:

You must create at least one SFTP server alias before creating an SFTP user alias.

➤ **To create an SFTP server alias**

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > SFTP**.
3. Click **Create Server Alias**.
4. Configure the following server alias settings:

Note:

Version 2 SFTP Client is introduced for PIE-59929 in IS_10.5_Core_Fix5.

Parameter	Specify
SFTP Server Alias Properties	
SFTP Client Version	<p>The SFTP client to use.</p> <p>Note: The Version 2 client has additional configuration properties, Key Exchange Algorithms, Machine Access Code (MAC) algorithms, and ciphers that are not available in the Version 1 client.</p>
Alias	<p>A name for the SFTP server alias.</p> <p>An SFTP server alias name:</p> <ul style="list-style-type: none">■ Can contain only underscores (_) and periods (.) as special characters.■ Cannot begin with the string "http://".■ Can be of a maximum length of 255 characters.
Host Name or IP Address	<p>The host name or IP address of the SFTP server.</p>
Port Number	<p>The port number of the SFTP server. The port number must be in the range of 0 to 65535 (inclusive).</p>
Proxy Alias	<p>The proxy alias through which requests should be routed. The proxy alias can be HTTP, HTTPS, or SOCKS. If a proxy alias is not specified, Integration Server makes outbound requests using each enabled proxy server alias until the request is sent successfully or all proxy servers have been tried. For more information about proxy aliases, see “Creating a Proxy Server Alias” on page 121.</p>
Host Key Location	<p>The location of the public key of the SFTP server. Integration Server validates the SFTP server using its public key.</p> <p>Important: The public key file must be present on the same machine on which you have installed Integration Server.</p>

Parameter	Specify
	<p>If you do not have the public key of the SFTP server, click Get Host Key. Integration Server retrieves the public key for the specified host and port and saves it in a temporary folder. Integration Server then displays the temporary folder path in the Host Key Location field.</p> <p>The version 2 SFTP client supports OpenSSH format host keys, and the existing key may be in the unsupported SSH2 format. So, you may not be able to save the server alias successfully. In such cases, regenerate the host key in the OpenSSH format, and click Get Host Key to get the regenerated key for the server alias.</p>

Note:

This change is applicable after installing IS_10.5_Core_Fix22, which upgrades maverick-client-1.7.23.jar to 1.7.34 for PIE-77771.

SFTP Server Alias Advanced Settings (Optional)

Min DH Key Size	The minimum DH key size. This parameter is not applicable to SFTP Client Version 1.
Max DH Key Size	The maximum DH key size. This parameter is not applicable to SFTP Client Version 1.
Preferred Key Exchange Algorithms	A list of key exchange algorithms that Integration Server presents to the SFTP server for key exchange. The algorithms are listed in the order of preference.
Preferred MAC Algorithms S2C	A list of Message Authentication Code (MAC) algorithms applicable to the messages sent from an SFTP server to the Integration Server. The algorithms are listed in the order of preference.
Preferred MAC Algorithms C2S	A list of Message Authentication Code (MAC) algorithms applicable to the messages sent from Integration Server to an SFTP server. The algorithms are listed in the order of preference.
Preferred Ciphers S2C	A list of preferred ciphers applicable to messages sent from an SFTP server to Integration Server .

Parameter	Specify
Preferred Ciphers C2S	A list of preferred ciphers applicable to messages sent from Integration Server to an SFTP server .

- To specify the order in which Integration Server presents the algorithms to the SFTP Server, select an algorithm, and click **Up** or **Down**.
- Every SFTP server supports a pre-defined set of algorithms and ciphers. The algorithm that both Integration Server and the SFTP server support is chosen.
- To exclude an algorithm from a list, select the algorithm, and click **Right**. The algorithm moves to the list of excluded algorithms.

For more information, see [“Upgrade Impact on Existing Server Alias Data”](#) on page 453.

5. Click **Save Changes**.

Note:

If you see the error message: "**com.maverick.ssh.SshException: Minimum DH p value not provided [2048]**", set **Min DH Key Size** to 2048.

Note:

Key Exchange Algorithms such as `diffie-hellman-group-exchange-sha1` and `diffie-hellman-group-exchange-sha256` consider values set for the **Min DH Key Size** and **Max DH Key Size** parameters. All other algorithms ignore these values.

Integration Server stores the SFTP server alias configuration along with the host key information in the `/<IntegrationServer_directory>/instances/<instance_name>/config/sftp/sftpServerAliases.cnf` file.

Editing an SFTP Server Alias

You can edit only the **Host Name or IP Address**, **Port Number**, and **Host Key Location** properties. You cannot leave the **Host Name or IP Address** and **Port Number** fields empty.

➤ To edit an SFTP server alias

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > SFTP**.
3. In the SFTP Server List, click the name of the SFTP server alias that you want to edit.
4. In the properties screen for the selected alias, make the necessary modifications.
5. Click **Save Changes**.

Creating an SFTP User Alias

An SFTP user alias is a named set of parameters that contains SFTP user account details and client configurations that Integration Server uses to function as an SFTP client.

In many organizations, a system administrator provides the SFTP user account information that you require to create an SFTP user alias.

You can have multiple SFTP user aliases for the same SFTP user account. Each SFTP user alias name in an Integration Server must be unique.

Keep the following points in mind when configuring Integration Server to act as an SFTP client:

- Integration Server supports password authentication and public key authentication for authenticating itself as the client to the SFTP server.
- For both password and public key authentication, you must have an account on the SFTP server that is set up for SFTP access.
- For public key authentication, the SFTP server and Integration Server must have access to their own private key and each other's public key.

Important:

You must create at least one SFTP server alias before creating an SFTP user alias.

➤ To create an SFTP user alias

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > SFTP**.
3. Click **User Alias Settings**.
4. Click **Create User Alias**.
5. Under **SFTP User Alias Properties**, provide the following information:

Parameter	Specify
Alias	<p>A name for the alias.</p> <p>An SFTP user alias name:</p> <ul style="list-style-type: none"> ■ Can contain only underscores (_) and periods (.) as the special characters. ■ Cannot begin with the string "http://". ■ Can be of a maximum length of 255 characters.

Parameter	Specify						
User Name	The user name for the SFTP user account.						
Authentication Type	<p>The type of authentication that Integration Server uses to authenticate itself to the SFTP server. Client authentication can be either by password or by public and private keys.</p> <table border="1"> <thead> <tr> <th>Select</th> <th>To</th> </tr> </thead> <tbody> <tr> <td>Password</td> <td>Use password authentication.</td> </tr> <tr> <td>Public Key</td> <td> <p>Authenticate Integration Server by using public and private keys.</p> <p>To use this authentication type, the SFTP server and Integration Server must each have access to their own private key and each other's public key. If you select public key authentication, Integration Server saves the private key file in the <i>Integration Server_directory/instances/instance_name/config/sftp/identities</i> folder.</p> </td> </tr> </tbody> </table>	Select	To	Password	Use password authentication.	Public Key	<p>Authenticate Integration Server by using public and private keys.</p> <p>To use this authentication type, the SFTP server and Integration Server must each have access to their own private key and each other's public key. If you select public key authentication, Integration Server saves the private key file in the <i>Integration Server_directory/instances/instance_name/config/sftp/identities</i> folder.</p>
Select	To						
Password	Use password authentication.						
Public Key	<p>Authenticate Integration Server by using public and private keys.</p> <p>To use this authentication type, the SFTP server and Integration Server must each have access to their own private key and each other's public key. If you select public key authentication, Integration Server saves the private key file in the <i>Integration Server_directory/instances/instance_name/config/sftp/identities</i> folder.</p>						
Password	If you are using password authentication, enter the password for the specified user to connect to the SFTP server.						
Re-type Password	Re-enter the above password.						
Private Key Location	If you selected Public Key as the authentication type, enter the location of the private key of the specified SFTP user.						
PassPhrase	If you selected Public Key as the authentication type and if the private key you specified requires a passphrase, enter the passphrase for the private key file of the specified user.						
Re-type PassPhrase	Re-enter the above passphrase.						
SFTP Server Alias	The alias of the SFTP server to which you want the specified user to connect.						
Maximum Retries	The number of times Integration Server attempts to connect to the SFTP server. The maximum allowed value is 6. The minimum allowed value is 1. The default is 6 retries.						
Connection Timeout (seconds)	The amount of time (measured in seconds) Integration Server waits for a response from the SFTP server before timing out and terminating the request. The default is 0, which indicates that the session will never time out.						

Parameter	Specify						
Session Timeout (minutes)	The number of minutes you want Integration Server to wait before terminating an idle session. The default is 10 minutes.						
Strict Host Key Checking	Whether Integration Server verifies the host key of the SFTP server before establishing a connection to the SFTP server.						
	<table border="0"> <thead> <tr> <th style="border-bottom: 1px solid black;">Select</th> <th style="border-bottom: 1px solid black;">To</th> </tr> </thead> <tbody> <tr> <td>Yes</td> <td>Allow Integration Server to verify the host key of SFTP server against the host key that was imported during the SFTP server alias configuration. If the host key is found to be the same, then Integration Server establishes a connection to SFTP server, else the connection to SFTP server fails.</td> </tr> <tr> <td>No</td> <td>Prevent Integration Server from verifying the host key of SFTP server during connection.</td> </tr> </tbody> </table>	Select	To	Yes	Allow Integration Server to verify the host key of SFTP server against the host key that was imported during the SFTP server alias configuration. If the host key is found to be the same, then Integration Server establishes a connection to SFTP server, else the connection to SFTP server fails.	No	Prevent Integration Server from verifying the host key of SFTP server during connection.
Select	To						
Yes	Allow Integration Server to verify the host key of SFTP server against the host key that was imported during the SFTP server alias configuration. If the host key is found to be the same, then Integration Server establishes a connection to SFTP server, else the connection to SFTP server fails.						
No	Prevent Integration Server from verifying the host key of SFTP server during connection.						
	<p>Note: For added security, Software AG recommends you to choose Yes and enable the host key verification.</p>						
Compression	Whether or not to compress the data to reduce the amount of data that is transmitted. Integration Server supports compression using the compression algorithm zlib.						
	<p>Note: You can use compression only if the SFTP server that you are connecting to supports compression.</p>						
	<table border="0"> <thead> <tr> <th style="border-bottom: 1px solid black;">Select</th> <th style="border-bottom: 1px solid black;">To</th> </tr> </thead> <tbody> <tr> <td>zlib</td> <td>Compress the data that is transmitted between the SFTP server and Integration Server.</td> </tr> <tr> <td>None</td> <td>Not compress the data.</td> </tr> </tbody> </table>	Select	To	zlib	Compress the data that is transmitted between the SFTP server and Integration Server.	None	Not compress the data.
Select	To						
zlib	Compress the data that is transmitted between the SFTP server and Integration Server.						
None	Not compress the data.						
Compression Level	The compression level to use if you selected the compression algorithm zlib in the Compression field. The minimum allowed value is 1 (fast, less compression). The maximum allowed value is 6 (slow, most compression). The default is 6.						

6. Click **Save Changes**.

Integration Server saves the SFTP user alias configuration in the *Integration Server_directory/instances/instance_name/config/sftp/sftpUserAliases.cnf* file.

Editing an SFTP User Alias

You can edit all the fields except the alias name and the user name for the SFTP user account.

> To edit an SFTP user alias

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > SFTP > User Alias Settings**.
3. In the SFTP User List, click the name of the SFTP user alias that you want to edit. Integration Server displays the properties screen for that alias.
4. In the properties screen for the selected alias, make the necessary modifications.
5. Click **Save Changes**.

Migration Impact on SFTP Configurations

Prior to Integration Server 9.12, **Preferred Key Exchange Algorithms** and **Proxy Alias** fields were specified in the SFTP user alias. These fields are now specified in the SFTP server alias. When you migrate to Integration Server 9.12 or later from an earlier version, Integration Server determines the values of the **Preferred Key Exchange Algorithms** and **Proxy Alias** fields as follows:

- If an SFTP server alias was never used in an SFTP user alias, Integration Server uses the default order for the **Preferred Key Exchange Algorithms** and the default value of `None` for the **Proxy Alias**.
- If an SFTP server alias was used in only one SFTP user alias, Integration Server migrates the order of **Preferred Key Exchange Algorithms** and the value of **Proxy Alias** from the SFTP user alias to the SFTP server alias.
- If an SFTP server alias was used in multiple SFTP user alias, Integration Server migrates the order of **Preferred Key Exchange Algorithms** and value of **Proxy Alias** from the first SFTP user alias associated with the SFTP server alias.

Upgrade Impact on Existing Server Alias Data

For Version SFTP client

The SFTP Version 1 client fields are populated with the values of the `watt.ssh.jsch.*` properties in the following manner:

- **Preferred Key Exchange Algorithms:** The algorithms included in the value of the `watt.ssh.jsch.kex` server property are added to "Preferred Key Exchange Algorithms", and all other algorithms are added to "Excluded Key Exchange Algorithms". If the value of the

watt property is empty, the default Key Exchange algorithms are added to "Preferred Key Exchange Algorithms".

- **Preferred MAC Algorithms S2C:** The algorithms included in the value of the `watt.ssh.jsch.mac_s2c` server property are added to "Preferred MAC Algorithms S2C" and all other algorithms are added to "Excluded MAC Algorithms S2C". If the value of the watt property is empty, then the default server-to-client MAC algorithms are added to "Preferred MAC Algorithms S2C".
- **Preferred MAC Algorithms C2S:** The algorithms included in the value of the `watt.ssh.jsch.mac_c2s` server property are added to "Preferred MAC Algorithms C2S" and all other algorithms are added to "Excluded MAC Algorithms C2S". If the value of the watt property is empty, then the default client-to-server MAC algorithms are added to "Preferred MAC Algorithms C2S".
- **Preferred Ciphers S2C:** The ciphers included in the value of the `watt.ssh.jsch.ciphers` server property are added to "Preferred Ciphers S2C" and all other ciphers are added to "Excluded Ciphers S2C".
- **Preferred Ciphers C2S:** The ciphers included in the value of the `watt.ssh.jsch.ciphers` server property are added to "Preferred Ciphers C2S" and all other ciphers are added to "Excluded Ciphers C2S".

For Version 2 SFTP client

- The latest Version 2 SFTP client supports OpenSSH format host keys, and the existing key may be in the unsupported SSH2 format. So, you may not be able to save the server alias successfully. In such cases, regenerate the host key in the OpenSSH format, and click **Get Host Key** to get the regenerated key for the server alias.
- The Preferred MAC Algorithms list excludes `hmac-sha256`, `hmac-sha256@ssh.com`, `hmac-sha512`, `hmac-sha512@ssh.com`, `hmac-ripemd160`, `hmac-ripemd160@openssh.com`, `hmac-ripemd160-etm@openssh.com` as they are not supported by the latest Version 2 SFTP client.

Note:

The upgrade impact for SFTP Version 2 SFTP client is applicable after installing `IS_10.5_Core_Fix22`, which upgrades `maverick-client-1.7.23.jar` to `1.7.34` for `PIE-77771`.

Note:

- All `watt.ssh.jsch.*` parameters, except `watt.ssh.jsch.logging`, are deprecated. Do not use the deprecated parameters because the preferred key exchange algorithms, ciphers, and MAC algorithms are configured from the user interface.
- Integration Server uses the `watt.ssh.jsch.logging` server configuration property to enable logging for both versions of the SFTP client.

Testing the Connection to the SFTP Server

After you add an SFTP user alias, you can test the connection to ensure that Integration Server can establish a connection with the SFTP server using the credentials and details you specified for the alias.

➤ **To test the connection to an SFTP server**

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > SFTP > User Alias Settings**.

Integration Server Administrator displays all the SFTP user alias definitions.

3. In the **Test** column for the alias that you want to test, click the ▶ icon.

Integration Server Administrator displays a status line that indicates whether or not the connection is successful.

If testing the user alias results in `java.security.NoSuchProviderException`, add the `org.bouncycastle.jce.provider.BouncyCastleProvider` security provider class in Integration Server Administrator > Security > Keystore > Add Security Provider.

24 Configuring Integration Server for Secure Communication

■ Overview	458
■ Anatomy of an Integration Server SSL Connection	458
■ Roadmap for Configuring SSL	460
■ Configuring Server-Side SSL Settings	463
■ Controlling Server SSL Security Level by Port	464
■ Storing SSL Information for the Integration Server JVM in a Secure Manner	464
■ Specifying Cipher Suites for Use with SSL	466
■ Usage of CA Certificates: Technical Considerations	467
■ WS-Security and Integration Server	468
■ Using SAML for Web Service Client Authentication	468
■ Requirements for Using SAML for Authentication	468
■ Identifying Trusted STSs to Integration Server	469
■ Accepting SAML2 Tokens at the Transport Level	471

Overview

An administrator can configure the Integration Server to use Secure Sockets Layer (SSL) to provide secure communications with the server. This chapter explains how SSL works with Integration Server, and the information that you need to configure SSL authentication for the server side.

For information about how Integration Server authenticates clients, and configuring the SSL client side, see [“Authenticating Clients” on page 515](#).

Anatomy of an Integration Server SSL Connection

It is useful to conceptualize an Integration Server SSL connection in terms of an SSL *server* and an SSL *client*. The request for an SSL connection originates from a client. The client can be one of the following:

- A partner application
- An Internet resource
- A web browser
- An Integration Server

During the SSL handshake process, the entity acting as the SSL server responds to the request for a connection by presenting its SSL credentials (an X.509 certificate) to the requesting client. If those credentials are authenticated by the client, either:

- An SSL connection is established and information can be exchanged between the client and server.
- or -
- The next phase of the authentication process occurs, and the server requests the SSL credentials of the client. If the server verifies those credentials (that is, the client's *identity*), an SSL connection is established and information exchange can take place.

Integration Server and SSL Connection Type

The types of SSL connection referred to above are termed *one-way* and *two-way* SSL authentication:

- In a *one-way* SSL connection, an anonymous client authenticates the credentials of a server in preparation for setting up a secure transaction. In most cases, the server knows nothing about the client's identity because verification of its credentials is not required. When desired, however the client can be authenticated by means such as basic username/password.

This type of authentication typifies connections where a browser establishes a connection to an Internet server to perform a secure transaction (for example, viewing a savings account, or buying items with a credit card). The client must authenticate the server's credentials before initiating the transaction, but it is not necessary (nor would it be practical) for the server to authenticate and keep a record of every possible client (browser).

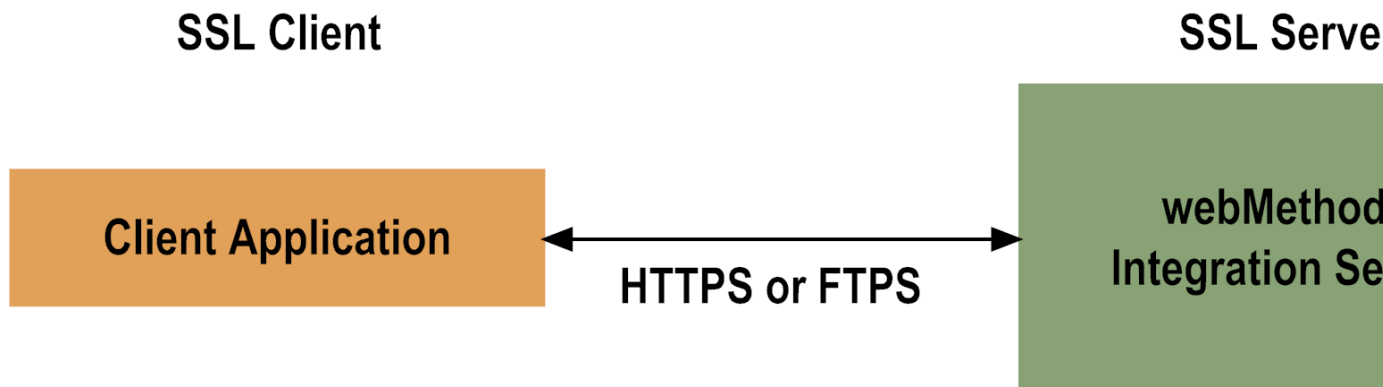
For Integration Server, this type of connection is typically one where a partner application or resource needs to verify the authenticity of the Integration Server without itself needing to be authenticated.

- In a *two-way* SSL connection, both client and server must authenticate each other's credentials before an SSL connection is established and information can be exchanged.

Unlike a one-way SSL connection, both Integration Server and the partner application or resource require access to each other's SSL certificates in order to authenticate each other, establish an SSL connection, and transmit information. Compared to a one-way connection, a two-way SSL connection provides a much higher level of security.

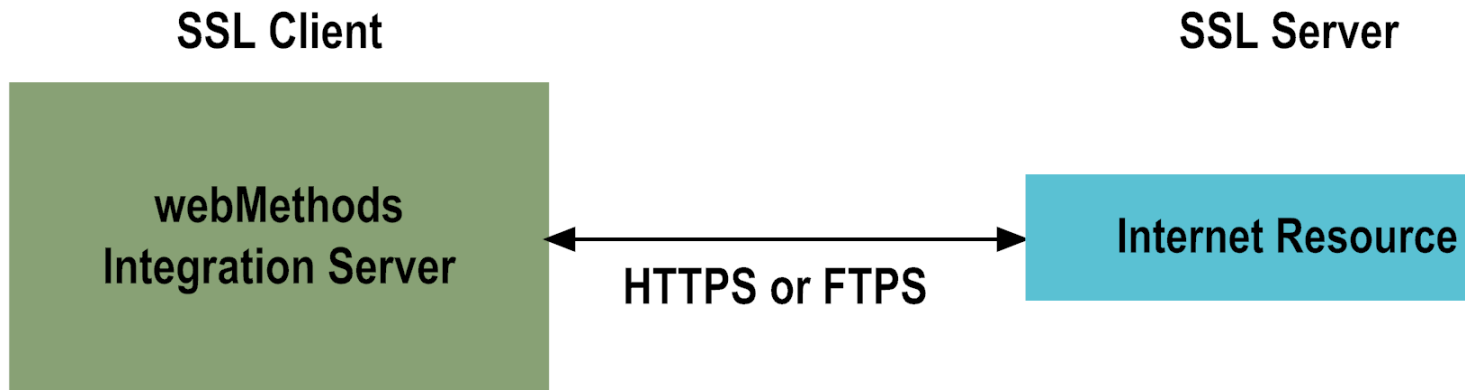
Integration Server as an SSL Server

When an IS client or partner application submits a request to Integration Server via HTTPS or FTPS, and a two-way SSL connection is established, the IS client acts as the SSL client and the Integration Server acts as the SSL server.



Integration Server as an SSL Client

When an Integration Server service submits an HTTPS or FTPS request to an Internet resource, the Integration Server is the SSL client and the Internet resource with which it is communicating acts as the SSL server.



Roadmap for Configuring SSL

The following table provides a high-level roadmap for configuring SSL on Integration Server.

Task	Activities	Notes
Create Integration Server keys and certificates	<ul style="list-style-type: none"> ■ Generate a public key/private key pair. ■ Generate a certificate signing request (CSR) and send to the certificate authority (CA) for signing. ■ Receive validated certificate from the CA. ■ Import signed certificate into a keystore. 	<p>Required for one-way and two-way SSL connections.</p> <p>Refer to the documentation for Java <i>keytool</i> or your certificate management tool.</p>
Create keystore and truststore for Integration Server	<ul style="list-style-type: none"> ■ Create a keystore and import the signed certificate and private key. ■ Create a truststore and import the certificate of the signing CA. ■ Store the keystore and truststore in a secure IS certificates directory. ■ Create aliases for the keystore and truststore. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Important: If you use Oracle <i>keytool</i> to create the keystore, you cannot import an existing private key. You can use other tools such as <i>OpenSSL</i> or <i>Portecle</i>.</p> </div>	<p>Required for one-way and two-way SSL connections.</p> <p>Refer to the following:</p> <ul style="list-style-type: none"> ■ The documentation for your certificate management tool ■ “Creating a Keystore” on page 478

Task	Activities	Notes
Obtain certificates of partner application or resource - and - Create certificate mapping	Use the Integration Server Administrator to save the following: <ul style="list-style-type: none"> ■ Signed certificate of the partner application. ■ Signed certificate of the CA for the partner's SSL certificate. 	Required for two-way SSL connections. Refer to the following: <ul style="list-style-type: none"> ■ “Importing a Certificate (Client or CA Signing Certificate) and Mapping It to a User” on page 519
Add an HTTPS or FTPS port (if none are defined)	If you want to allow only secure connections to the server: <ul style="list-style-type: none"> ■ Ensure that the primary port uses an HTTPS port. ■ Delete all other non-HTTPS ports. Add additional HTTPS or FTPS ports as required.	Required for one-way and two-way SSL connections. Refer to “Configuring Ports” on page 149 .

Creating Integration Server Keys and Certificates

Use a standard certificate management tool, such as OpenSSL or Portecle, to generate a private/public key pair for Integration Server. Then, place the public key in a certificate signing request (CSR).

After creating the CSR, send to the CA to sign the CSR. Request the certificate in DER format. If you receive a certificate in PEM format (or any format other than DER), you need to convert it to DER format.

The signing CA's certificate attests to the identity of the CA that signed the digital certificate for the Integration Server. The CA should send this certificate to you when it sends you the digital certificate for the Integration Server.

Once you receive your signed certificate from the CA, you need to import the certificate into a keystore. You will then have an SSL certificate and private key to use with Integration Server.

In general, you will repeat the steps after creation of the key pair about every year or two years, at the time you need to renew the certificate.

If certificates contain certificate extensions that you want Integration Server to validate, set the `watt.security.cert.wmChainVerifier.enforceExtensionsChecks` server configuration property to `true`.

Note:

The above process is described in general terms. The procedures may vary somewhat, depending upon the CA that you use.

Creating a Keystore and Truststore

Keystores and truststores are files that function as repositories for storage of keys and certificates necessary for SSL authentication, encryption/decryption, and digital signing/verification services. Keystores and truststores provide added layers of security and ease of administration, compared to maintaining the keys and certificates in separate files.

For information about creating keystores and truststores, importing keys and certificates into keystores and truststores, and other operations with these files, refer to the documentation for your certificate management tool.

For information about using Integration Server with keystores and truststores and how to create aliases for these files, see [“Using Keystores and Truststores with Integration Server”](#) on page 477.

Obtaining the Certificates and Keys of the Partner Application

If your Integration Server will run services that submit HTTPS or FTPS requests to other resources on the Internet, the Integration Server will be acting as a client and will receive certificates from these resources. In order for these transactions to work, your Integration Server must have copies of their public keys and signing CA certificates. Refer to [“Authenticating Clients”](#) on page 515 for information on importing client certificates to Integration Server and setting up client authentication.

Configuring an HTTPS or FTPS Port

Integration Server requires an HTTPS or FTPS port for SSL connections.

If you want to allow only secure connections to the server, ensure that the primary port uses an HTTPS or FTPS port and delete all other non-HTTPS or non-FTPS ports. Add as many additional HTTPS or FTPS ports as you want. Make sure no other applications are listening on the port you want to use.

You can further configure HTTPS and HTTPS diagnostic ports and the `pub.client:http` service, or FTPS ports and the `pub.client:ftp` service to use Java Secure Socket Extension (JSSE) for handling inbound and outbound communications. JSSE is required to support TLS 1.1 or 1.2. When configured, Integration Server uses the JSSE socket factory to create outbound socket connections. For more information about configuring ports to use JSSE, see [“Configuring Ports”](#) on page 149. For more information about setting JSSE in the `pub.client:http` service and `pub.client:ftp` service, see *webMethods Integration Server Built-In Services Reference*.

Note:

When you configure a port to use JSSE and a client presents a certificate that Integration Server does not trust, Integration Server rejects the request and it will not fall back to use basic credentials (user name/password). If there is no certificate mapping, Integration Server issues an invalid credentials error.

For HTTPS protocol, the standard port is 443.

For information about configuring an HTTPS or FTPS port, see [“Configuring Ports”](#) on page 149.

Configuring Server-Side SSL Settings

The configuration settings covered in this section deal with the Integration Server SSL *server* side. Configuring SSL for Integration Server *clients* entails mapping their X.509 certificates to Integration Server users. This information is covered in [“Customizing Authentication Using JAAS” on page 527](#).

Note:

The SSL certificate settings described in this section are also used for Integration Server web services, and for built-in services used to secure Integration Server XML documents. For more information, refer to the *Web Services Developer's Guide* and the *webMethods Integration Server Built-In Services Reference*.

Before configuring SSL for Integration Server, you must have previously used your certificate management tool to do the following:

- Create at least one keystore, in JKS or PKCS12 format, containing an Integration Server key pair to use for SSL and its corresponding key alias.
- Create at least one truststore, in JKS format, containing the trusted root certificate of the signing CA (and certificate chain, if necessary).
- Create a keystore and a truststore alias.

Specifying the Integration Server SSL Authentication Credentials

The Integration Server SSL configuration settings are organized into several groups. You can select a **Keystore Alias** and **Key Alias** for the following groups of settings:

- **SSL Key**, which specifies the Integration Server private and public key pair to use when presenting Integration Server's SSL credentials to a requesting partner application, Internet resource, or web service.
- This setting determines the Integration Server's SSL *identity*.
- **Signing Key**, which specifies the private key with which to sign outgoing documents, messages, and data streams from Integration Server.
- **Decryption Key**, which specifies the private key to use for decrypting incoming documents, messages, and data streams from external sources, where the information was encrypted with the associated Integration Server public key.

For the *Truststore*, which specifies the location of the signing CA certificates for SSL authentication, you specify its **Truststore Alias**.

➤ To configure the server for SSL authentication

1. Open the Integration Server Administrator if it is not already open.

2. In the **Security** menu of the Navigation panel, click **Certificates**.
3. Click **Edit Certificates Settings**.
 - Select a **Keystore Alias** and **Key Alias** for the **SSL Key, Signing Key,** and **Decryption Key**.
 - Select a **Truststore Alias** for the **Truststore**.

Important:

The settings on this screen are the *default* SSL values used to identify the Integration Server, and specify the SSL keys to use with any Integration Server document, web service, or built-in service. Do not change these values without first consulting with your system administrator or security administrator.

4. Click **Save Changes**.

Controlling Server SSL Security Level by Port

You can configure your Integration Server to present different server certificates with different ports. One reason to do this is so that different ports can provide different SSL security levels. You determine the security level of a certificate during the certificate signing process. You tell the certificate authority which class of certificate you need and it creates a certificate with those attributes. Later, when you configure your ports, you specify the certificate that has the security level you want to associate with that port. See [“Setting Up Aliases for Remote Integration Servers” on page 115](#) for instructions on configuring your ports.

Storing SSL Information for the Integration Server JVM in a Secure Manner

When using secure socket layers (SSL) with external servers, you need to configure JVM parameters for creating SSL Context for the SSL handshake. You can set the keystore location, truststore location, and password information using the `javax.net.ssl` properties for the JVM. However, these properties take String values which can result in storing password information in plain text somewhere on the file system. This represents a security vulnerability.

To address this security issue, Integration Server provides a way to specify the keystore and truststore locations and passwords in a way in which the passwords are not stored in plain text. Specifically, Integration Server provides server configuration parameters for which you specify the keystore and truststore aliases to be used to establish the default SSL context. At start up, Integration Server sets the `javax.net.ssl` properties by obtaining the store locations and passwords from the aliases and then creates the default SSL context.

The server configuration parameters are as follows:

- `watt.server.ssl.keyStoreAlias` - Name of the keystore alias for the Integration Server keystore that contains the information needed to establish an SSL connection with SSL-enabled servers.

- `watt.server.ssl.trustStoreAlias` - Name of the truststore alias for the Integration Server truststore that contains the information needed to establish an SSL connection with SSL-enabled servers.

➤ **To store the SSL information for the JVM used by Integration Server in a secure manner**

1. Create the keystore to use for creating the default SSL context.

Note: Software AG does not provide its own set of utilities for creating or managing keystore or truststore files.

2. If needed, create a truststore to use for creating the default SSL context.
3. Use Integration Server Administrator to create a keystore alias for the keystore created in step 1.

For more information about creating a keystore alias, see [“Creating Keystore Aliases” on page 481](#).

4. If necessary use Integration Server Administrator to create a truststore alias for the truststore created in step 2.

For more information about creating a truststore alias, see [“Creating Truststore Aliases” on page 482](#).

5. Use Integration Server Administrator to set the `watt.server.ssl.keyStoreAlias` parameter value to be the keystore alias created in step 3.
6. If you are using a truststore, use Integration Server Administrator to set the `watt.server.ssl.trustStoreAlias` parameter value to be the truststore alias created in step 4.
7. Restart Integration Server.

Important:

If you change the value of the `watt.server.ssl.keyStoreAlias` parameter or the `watt.server.ssl.trustStoreAlias` parameter, you must restart Integration Server for changes to take effect. Additionally, if you change the contents of the keystore and/or truststore referenced by the parameters, you must restart Integration Server for changes to take effect.

Order of Precedence for the `javax.net.ssl` Properties

Integration Server can set the `javax.net.ssl` properties in the JVM can be set in various ways. For example, you can modify the `custom_wrapper.conf` file such that Integration Server sets the properties at start up. Additionally, you can use server configuration parameters and a package start-up services to set the properties. Integration Server uses the following precedence order to set the values of the `javax.net.ssl` properties in the JVM.

1. Package start-up service.

2. The `watt.server.ssl*` server configuration parameters, specifically
 - `watt.server.ssl.keyStoreAlias`
 - `watt.server.ssl.trustStoreAlias`
3. The `watt.config.systemProperties` server configuration parameters.
4. `custom_wrapper.conf` (Integration Server)
 - `JAVA_CUSTOM_OPTS` in `Integration Server_directory/bin/setenv.bat` (Microservices Runtime)

Important:

Any change that impacts the value of the `javax.net.ssl` properties requires a restart of Integration Server for the changes to take effect. For example, if you do any of the following, you must restart Integration Server:

- Change the keystore whose alias is used by the `watt.server.ssl.keyStoreAlias` parameter.
- Change the truststore whose alias is set in the `watt.server.ssl.trustStoreAlias` parameter.
- Change the alias used by `watt.server.ssl.keyStoreAlias` or `watt.server.ssl.trustStoreAlias`.
- Edit `custom_wrapper.conf` or `JAVA_CUSTOM_OPTS` to change the value of the `javax.net.ssl` parameters.

Specifying Cipher Suites for Use with SSL

Integration Server provides server configuration parameters that you can use to specify the cipher suites that can be used with inbound and outbound SSL requests.

Server Configuration Parameter	Description
<code>watt.net.jsse.client.enabledCipherSuiteList</code>	Specifies the cipher suites used on JSSE sockets that are used while making outbound HTTPS or FTPS requests.
<code>watt.net.jsse.server.enabledCipherSuiteList</code>	Specifies the cipher suites used on Integration Server ports that use JSSE and handle inbound requests.
<code>watt.net.ssl.client.cipherSuiteList</code>	Specifies the cipher suites for outbound SSL connections.
<code>watt.net.ssl.server.cipherSuiteList</code>	Specifies the cipher suites for inbound SSL connections.

While the above parameters use a comma-separated list to identify the allowed cipher suites, you can also use a file as the value for any of the parameters. Using a file can make it easier to specify a long list of cipher suites.

Keep the following information in mind when using a file to specify the allowed cipher suites:

- In the file, specify each cipher suite on a different line.

- For each cipher suite server configuration property for which you want to specify a file instead of a list of cipher suites, specify the following as the value of the property:

```
file:directoryName\filename
```

For example: `watt.net.jsse.server.enabledCipherSuiteList=file:c:\ssl\ciphers.txt`

- Integration Server loads the file and its list of supported cipher suites at start up. Changes to the contents of the file that are made after Integration Server starts will not take effect until the next time Integration Server starts.
- You can set the value of a cipher suite server configuration parameter to a comma-separated list, default, or the absolute path to a file. You cannot specify a combination of these for a single parameter.

Usage of CA Certificates: Technical Considerations

Following are technical considerations for usage of CA certificates and trusted certificate directories.

Handling Expired CA Certificates

At times, one or more CA certificates in a chain may pass their expiration dates. When a web browser connects to the Internet resource having such an expired certificate, it might still accept the connection. The web browser can accept the connection if it can access a valid certificate for the CA with the expired certificate. Integration Server, however, *cannot* connect to a resource with an expired signing certificate in the chain, unless explicitly configured to do so.

In certain cases, you may want Integration Server to accept a connection when one or more of its CA certificates are expired. In cases such as these, you may need to change a server configuration property. For more information, refer to [“Working with Extended Configuration Settings” on page 127](#). Remember to restart the server after changing the setting.

Note:

It is less secure to ignore the expired certificates than to deny the connection because of expired certificates.

Customizing Usage of the Trusted Certificates Directory

Most of the time you will want to specify a trusted certificates directory; however, there may be times when you do not want to. For example, you might want to trust all certificate authorities on outbound requests and trust specific CAs on different ports for incoming requests. For outbound requests (a certificate the server receives from a server that it submits a request to), if you do not specify a directory, or specify a directory that does not contain certificates for CAs, by default, the server trusts all certificate authorities. The server configuration property that controls this behavior (`watt.security.cert.wmChainVerifier.trustByDefault`) is set to `True` by default. If this property is set to `False` and no directory or an empty directory is specified, the server will trust no certificates for outbound requests.

For inbound requests, you can specify a trusted certificates directory at the server level (on the Security Certificates screen) or at the port level (on the Edit HTTPS Port Configuration screen or the Edit FTPS Port Configuration screen). If you omit a trusted authorities directory (or specify a directory that does not contain CA certificates) from both the server level and the port level, the server will trust no certificate authorities. If you specify a trusted authorities directory at the server level and at the port level, the server uses the directory specified at the port level for determining trust on connections being made to that port. If you specify a trusted authorities directory at just the port level, the server uses the port-level setting for requests being made to the port.

For S/MIME signature trust validation, if you do not specify a trusted certificates directory or specify a directory that does not contain the certificates of trusted CAs, by default the server trusts all signatures on S/MIME messages. However, if `watt.security.cert.wmChainVerifier.trustByDefault` is set to `False` and no directory or an empty directory is specified, the server will trust *no* signatures on S/MIME messages.

WS-Security and Integration Server

Integration Server supports the WS-Security standard for use with web services. In contrast to transport-based authentication frameworks such as HTTPS and FTPS, which secure the endpoints of a connection against threats, WS-Security secures the message transmission environment *between* endpoints. Authentication information, which is included in the SOAP message header, can be saved in X.509 certificates, user name tokens or SAML tokens, and may include the actual certificates or references.

The focus of this chapter is on the use of transport-based security with Integration Server. However, when configuring an Integration Server web service for WS-Security, you specify its SSL, signing, and decryption keys with Integration Server Administrator the same as you would for setting up transport-based security (see [“Specifying the Integration Server SSL Authentication Credentials”](#) on [page 463](#)). If you intend to use WS-Security with your Integration Server-based web services, see *webMethods Service Development Help* and the Oasis Standards documentation for WS-Security.

Using SAML for Web Service Client Authentication

If you want to use policies based on `WS-SecurityPolicy` for authentication and those policies require SAML tokens, you must set up Integration Server so that it can process the SAML tokens. Integration Server supports SAML tokens only in policies attached to provider web service descriptors for inbound requests.

For an inbound request message received by a provider web service, Integration Server must be able to validate the SAML token using the Java Authorization and Authentication Service (JAAS) login modules of Integration Server.

Requirements for Using SAML for Authentication

The following table lists the requirements that must be met so that Integration Server can process SAML tokens in policies based on `WS-SecurityPolicy` and for sending SAML2 tokens in the HTTP header.

Requirement	Description
Security Token Service (STS) provider	<p>Determine which STSs you want Integration Server to trust. Clients can use any STS provider that generates SAML 1.0 or 2.0 tokens. The generated SAML token must:</p> <ul style="list-style-type: none"> ■ Contain the client certificate if Integration Server is to process Holder-of-Key (HOK) type SAML assertions. Integration Server uses the client certificate to identify the client and map the client to an Integration Server user. ■ Be signed by the STS.
Certificates for each possible issuer of SAML assertions	Create a truststore that contains the public keys of each STS. For more information about creating a truststore, see “Creating a Keystore and Truststore” on page 462 .
Identification of trusted issuers	Identify trusted STSs to Integration Server. For instructions, see “Identifying Trusted STSs to Integration Server” on page 469 .
Client certificate mapping	If Integration Server is to process Holder-of-Key (HOK) type SAML assertions, which contain the public key of the client, you must map the client’s public key to an Integration Server user. For more information about configuring client certificates, see “Client Certificates” on page 518 .

Identifying Trusted STSs to Integration Server

If you want to use policies based on WS-SecurityPolicy that include SAML tokens for client authentication or accept SAML2 assertions through the HTTP header, you must set up Integration Server so that it can process the SAML tokens. One of the requirements is to identify STSs you want Integration Server to trust.

➤ To identify a trusted STS to Integration Server

1. In Integration Server Administrator, go to **Security > SAML**.
2. Click **Add SAML Token Issuer**.
3. Provide information in the following fields:

Parameter	Specify
Issuer Name	Name of a SAML token issuer from which Integration Server should accept and process SAML assertions. This value must match the value of the Issuer field in the SAML assertion.

Parameter	Specify
	<p>Integration Server will reject SAML assertions from issuers not configured on this screen and will log a message similar to the following to the Server log:</p> <pre data-bbox="513 373 1362 464">2010-06-09 23:35:38 EDT [ISS.0012.0025E] Rejecting SAML assertion from issuer "SAMPLE_STS" because issuer is not configured on the Security > SAML screen.</pre>
Truststore Alias	<p>A text identifier for the truststore, which contains the public keys of the SAML token issuer. Integration Server populates the Truststore Alias list with the existing truststore aliases.</p>
Certificate Alias	<p>A text identifier for the certificate associated with the truststore alias. Integration Server populates the Certificate Alias list with the certificate aliases from the selected truststore alias.</p>
Clock Skew	<p>Clock difference, in milliseconds, between the machine that hosts Integration Server and the SAML token issuer. Specify a non-negative number.</p> <p>For example, if the clock on the Integration Server machine and the issuer clock have 3 seconds time difference, you could specify a skew of 3001 milliseconds. To allow for some buffer, you could specify a slightly higher skew such as 3200 or even 4000.</p> <p>After parsing the SAML Assertion, Integration Server converts the timestamps into milliseconds and performs all validations using the milliseconds. As a result, SAML validation performed by Integration Server supports the use of different time zones for an issuer and Integration Server.</p> <p>When validating the NotBefore claim, Integration Server subtracts the clock skew from the NotBefore time found in the assertion (where the timestamp is now expressed in milliseconds). Then Integration Server compares the adjusted NotBefore time to the current time on the machine that hosts Integration Server to verify that the time on the Integration Server machine is lower than the adjusted NotBefore time.</p> <p>When validating the NotAfter claim, Integration Server subtracts the clock skew from the current time (as expressed in milliseconds). Then Integration Server compares the adjusted current time to the NotAfter time found in the assertion to ensure that the NotAfter time is greater than or equal to the adjusted current time.</p>

4. Click **Save Changes**.

Accepting SAML2 Tokens at the Transport Level

Integration Server supports including SAML2 tokens in the HTTP header, making it possible to use SAML2 tokens with all types of services. This enables integration with other security providers.

To use this functionality, you must do the following on Integration Server:

1. Configure the Integration Server `is_jaas.cnf` as shown below by adding the `SamlAssertLoginModule` as the first login module to the `IS_Transport` login context:

```
com.wm.app.b2b.server.auth.jaas.SamlAssertLoginModule requisite mode="transport"
defaultUserName="Default";
```

2. Add the issuer of the SAML assertion to the list of trusted SAML issuers. For more information, see [“Identifying Trusted STSs to Integration Server”](#) on page 469.

The client sending the request must include the custom HTTP header named `"wmIS-SAML2-Assertion"` and send the Base64 encoded SAML2 assertions as the header value.

When Integration Server receives an HTTP request with the custom header `"wmIS-SAML2-Assertion"` and finds a Base64 encoded SAML2 assertion in the header, Integration Server decodes from Base64 and validates the assertion. If validation of the assertion succeeds, Integration Server searches for an Integration Server user that matches the `NameID` from the SAML2 Assertion. Integration Server first checks for a local user defined on Integration Server and then searches Central Users or LDAP. If Integration Server finds a username that matches the `NameID`, Integration Server uses that username for the session. Otherwise, Integration Server uses the user defined in the `"defaultUserName"` option of the `SamlAssertLoginModule` module shown above. If the `"defaultUserName"` option is set to `"Default"`, Integration Server uses the Default user account, which allows access to resources that have the Anonymous ACL.

25 Setting Up SSL Session Logging

■ Overview of the SSL Session Log	474
■ Enabling the SSL Session Logging	475
■ Viewing the SSL Session Log	475
■ Changing the Default SSL Session Log Location	475
■ Managing the SSL Session Log Size	476
■ Avoiding Duplicate Entries in the Log	476

Overview of the SSL Session Log

SSL session log contains SSL session information in JSON format for inbound connections. It contains information on the cipher suite used, selected protocol version, and client details along with server and session creation details. Using this information, you can analyze the details of a successful SSL handshake.

Integration Server captures the SSL session information for Entrust and JSSE security providers. The following are excerpts from sample SSL session logs for JSSE and Entrust.

```
2019-07-15 11:21:49 IST {
  "provider" : "JSSE",
  "loggedInUser" : "Administrator",
  "sessionID" : "[93, 44, 20, 115, 150, 122, 228, 76, 181, 94, 62, 91, 207, 251, 222,
105, 91, 14,
  208, 203, 34, 115, 100, 112, 123, 71, 130, 212, 150, 225, 7, 137]",
  "serverPort" : 12346,
  "creationTime" : 1563169907869,
  "lastAccessedTime" : 1563169909320,
  "selectedCipherSuite" : "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
  "selectedProtocol" : "TLSv1.2",
  "clientIPAddresses" : "127.0.0.1",
  "clientPort" : 64782
}
```

```
2019-07-15 11:21:34 IST {
  "provider" : "Entrust",
  "loggedInUser" : "Administrator",
  "sessionID" : "5D:FC:C7:77:19:ED:4B:B2:94:9A:DB:4D:51:9B:CD:96",
  "serverPort" : 12345,
  "creationTime" : 1563169893970,
  "lastAccessedTime" : 1563169894006,
  "selectedCipherSuite" : "TLS_RSA_WITH_AES_256_CBC_SHA",
  "selectedProtocol" : "TLSv1.0",
  "clientIPAddresses" : "127.0.0.1",
  "clientPort" : 64775
}
```

Note:

These sample logs are configured to prettyPrint and include timestamps. For more information on configuring prettyPrint and timestamp, see [“Viewing the SSL Session Log” on page 475](#).

The following table describes the fields present in the SSL session log file:

Fields	Description
provider	Identifies the security provider used for the SSL session. Integration Server supports JSSE and Entrust security providers for SSL sessions.
loggedInUser	Specifies the user name of the client who initiated the SSL connection.
sessionID	Specifies the session ID created during the SSL handshake.
serverPort	Specifies the port that received the request.

Fields	Description
creationTime	Specifies SSL session creation time.
lastAccessedTime	Specifies the time when the SSL session was last used.
selectedCipherSuite	Specifies the cipher suite used for the SSL handshake.
selectedProtocol	Specifies the protocol version used for the SSL handshake.
clientIPAddresses	Specifies the IP address of the client.
clientPort	Specifies the client port number used for establishing an SSL connection.

Enabling the SSL Session Logging

To enable or disable SSL session logging, Integration Server provides a server configuration parameter. When you set the value of this parameter to true, Integration Server logs the SSL session information in the `inboundSSLSessions.log`.

For more information about the `watt.net.ssl.server.sessionlog` server configuration parameter, see the parameter description in [“Server Configuration Parameters” on page 1017](#).

Viewing the SSL Session Log

To view the SSL session log, go to `Software AG_directory\Integration Server_directory\instances instance_name \logs` directory and open the `inboundSSLSessions.log` file in a text editor. However, you can change the default location of the SSL session log file.

Using the `watt.net.ssl.server.sessionlog.prettyPrint` parameter, you can format the `inboundSSLSessions.log` file. When you set the value of this parameter to true, Integration Server formats the JSON with carriage returns and indentation to ease readability.

To include a timestamp in the SSL session log entries, set the value of `watt.net.ssl.server.sessionlog.includeTimestamp` parameter to true.

For more information about `watt.net.ssl.server.sessionlog.prettyPrint` and `watt.net.ssl.server.sessionlog.includeTimestamp` server configuration parameters, see the parameter description in [“Server Configuration Parameters” on page 1017](#).

Changing the Default SSL Session Log Location

You can change the default SSL session log location, including the name of the file to which the logging data is written. The `watt.net.ssl.server.sessionlog.file` server configuration parameter controls the location and name of the log file.

By default, the SSL session log location is: `Software AG_directory\Integration Server_directory\instances\instance_name \logs\inboundSSLSessions.log`

To change the SSL session log location or file, modify the `watt.net.ssl.server.sessionlog.file` parameter to specify the absolute path or relative path to the file to which Integration Server writes the SSL session log information. The relative path is relative to the Integration Server home directory: *Integration Server_directory\instances\instance_name*.

You must specify a path with a directory name and a filename. Restart Integration Server for changes to take effect.

Managing the SSL Session Log Size

By default, Integration Server writes the SSL session information to `inboundSSLSessions.log`. If the number of incoming HTTPS requests increase, then `inboundSSLSessions.log` can increase rapidly in size. In addition to consuming resources, a large `inboundSSLSessions.log` file is difficult to examine. To avoid this, you can configure Integration Server to rotate the `inboundSSLSessions.log` file based on the file size.

Using the `watt.net.ssl.server.sessionlog.maxFileSize` server configuration parameter, you can specify the size limit of the SSL session log. When `watt.net.ssl.server.sessionlog.maxFileSize` is set to a valid value, Integration Server rotates the `inboundSSLSessions.log` file by renaming the current log and creating a new `inboundSSLSessions.log` file. Integration Server renames the file to `inboundSSLSessions_<DATE(YYYYMMDD)>_TIME(HHMMSS).log` and creates a new `inboundSSLSessions.log` file.

For more information about the `watt.net.ssl.server.sessionlog.maxFileSize` server configuration parameter, see the parameter description in [“Server Configuration Parameters” on page 1017](#).

Avoiding Duplicate Entries in the Log

You can configure Integration Server to avoid duplicate entries in the SSL session log by caching session log entries. The `watt.net.ssl.server.sessionlog.cacheLogEntries` server configuration parameter instructs Integration Server to track the SSL session log entries in cache. When you set the value of this parameter to `true`, Integration Server writes an SSL session entry to cache. If a session is reused and exists in the cache, then Integration Server does not log the SSL session information for that session again. This eliminates duplicate entries in the log file.

However, you can set the duration for which SSL session entries remain in the cache using the `watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime` server configuration parameter. An SSL session entry in the cache expires after the time you set for `watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime` server configuration parameter. Integration Server removes the expired SSL session entries from its cache. If a client reuses a session for which Integration Server has removed the log entry, then Integration Server logs the session information again for that session.

Integration Server uses a sweeper task called SSL Session Log Entries Sweeper to remove the expired sessions.

For more information about the `watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime` server configuration parameter, see the parameter description in [“Server Configuration Parameters” on page 1017](#).

26 Using Keystores and Truststores with Integration Server

■ Keystores and Truststores	478
-----------------------------------	-----

Keystores and Truststores

Integration Server stores its private keys and SSL certificates in keystore files and the trusted roots for the certificates in truststore files. Keystores and truststores are secure files with industry-standard file formats.

Keystore File

Integration Server uses a special file called a *keystore* to store SSL certificates and keys.

A keystore file contains one or more pairs of a private key and signed certificate for its corresponding public key. The keystore should be strongly protected with a password, and stored (either on the file system or elsewhere) so that it is accessible only to administrators.

Keystore File Formats

The default, certificate file format for an Integration Server keystore is JKS (Java keystore), the proprietary keystore implementation provided by Oracle. You can also use PKCS12, a commonly used, standardized, certificate file format that provides a high degree of portability. Other keystore types can be made available by:

- Loading additional security providers
- Setting the `watt.security.keystore.supportedTypes` property.

HSM-Based Keystores

Under certain conditions, Integration Server supports the use of keystore files stored on a Hardware Security Module (HSM). Integration Server supports HSM-based keystores for ports, but not for other components. You cannot use HSM-based keystores with remote server aliases, outbound certificates, an internal server port, WS-Security, or Integration Server public services.

Only nCipher hardware card modules are currently supported.

Creating a Keystore

You can create and manage Integration Server keystores at the command line using `keytool`, the Oracle Java certificate editor.

You can also use other standard tools such as OpenSSL and Portecle.

Note: Software AG does not provide its own set of keystore utilities for creating or managing keystore files.

Truststore File

Integration Server uses a *truststore* to store its trusted root certificates, which are the public keys for the signing CAs. Although a truststore can contain the trusted roots for entire certificate chains,

there is no requirement for the organization of certificates within an Integration Server truststore. It simply functions as a database containing all the public keys for CAs within a specified trusted directory.

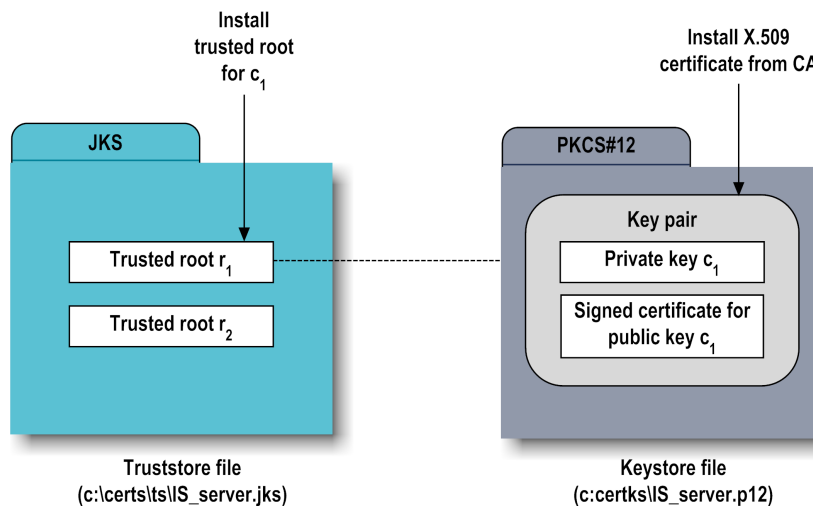
Truststore File Formats

The default format for a truststore is Java keystore (JKS). JKS is the proprietary keystore implementation provided by Oracle. You can create and manage JKS truststores at the command line using *keytool*, the Oracle Java certificate editor.

How Integration Server Uses a Keystore and Truststore

For an Integration Server component to be SSL authenticated, it must have a valid, authorized X.509 certificate installed in a keystore file *and* the private key and signing certificate for the certificate issuer (CA) installed in a truststore file. The following figure illustrates these requirements and the relationship between the two files.

Example Truststore File and Keystore File Showing Relationship



As shown in the above figure, the same truststore file can contain multiple trusted root certificates (public keys for the signing CAs). These trusted roots might be associated with numerous keystore files. A keystore file can contain the key pairs for multiple Integration Server components, and the entire certificate chain required for a component's authentication.

With a certificate chain, it is necessary to validate each subsequent signature in the list until a trusted CA certificate is reached. For Integration Server, you must include the entire chain of certificates in a keystore and truststore. Also, any root CA certificates in use by clients must be in an Integration Server truststore.

Protecting Keystore and Truststore Files

Keystore and truststore files exist within the file system of your operating system, and since these are critically important files, you want to maintain them in a secure directory path. If either of the

these files cannot be located, Integration Server authentication will be disabled and no connection with the server can be made. It is recommended that only users serving as My webMethods or Integration Server administrators have access to these certificate files.

Keystore, Truststore, and Key Aliases

To identify a particular keystore or truststore file, or private key within a keystore, Integration Server uses aliases. This use of aliases is the same as that by Java *keytool* and other certificate management tools. The use of aliases simplifies keystore and truststore management, because you do not need to enter path information when specifying a keystore, truststore, or private key in Integration Server Administrator or when using Integration Server public services.

You can view, create, and edit keystore and truststore aliases from the **Security > Keystore** panel in Integration Server Administrator.

Note:

Keystore and truststore aliases are *not* case-sensitive.

A *key alias* is a label for specific key within a keystore. Key aliases are created using your third-party certificate management tool. Although you do not create key aliases in Integration Server Administrator, you will need to identify which keys to use for SSL authentication, signing, and decrypting, by selecting the appropriate key aliases in Integration Server Administrator.

Default Keystore and Truststore Aliases

Integration Server creates default keystore and truststore aliases the first time Integration Server starts.

Important:

Use these aliases only in development environments where testing and debugging is performed. Do not use these aliases in a production environment.

The following table lists the predefined keystore and truststore aliases for Integration Server.

Default Alias	Description
DEFAULT_IS_KEYSTORE	The name of the default keystore alias for Integration Server. The DEFAULT_IS_KEYSTORE points to the <i>Software AG_directory</i> /common/conf/keystore.jks file and the default password is “manage” for all the entries.
DEFAULT_IS_TRUSTSTORE	The name of the default truststore alias for Integration Server. The DEFAULT_IS_TRUSTSTORE points to <i>Software AG_directory</i> /common/conf/platform_truststore.jks file and the default password is “manage” for all the entries.

You can edit and delete the default aliases. The certificate file format of the default keystore and truststore is Java keystore (JKS). The provider for the default keystore and truststore alias is the one shipped with the JVM.

Creating Keystore Aliases

The following procedures shows how to assign aliases to keystore files that you have created with the Oracle Java keytool or with another third-party certificate tool.

➤ To create an alias for a keystore file

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Keystore**.
3. Click **Create Keystore Alias**.
4. Enter the **Keystore Properties** settings as follows:

For this setting	Specify
Alias	<p>A text identifier for the keystore file.</p> <p>The keystore contains the private keys and certificates (including the associated public keys) for an Integration Server, partner application, or Integration Server component.</p>
Description	Optional. A text description for the keystore alias.
Type	<p>The certificate file format of the keystore file, which by default is JKS for keystores. You can also use PKCS12 format for a keystore.</p> <p>Other keystore types can be made available by:</p> <ul style="list-style-type: none"> ■ Loading additional security providers. ■ Setting the <code>watt.security.keyStore.supportedTypes</code> server configuration parameter.
Provider	<p>The provider that is used for the keystore or truststore type. The default provider is the one shipped with the JVM, which can be Oracle, IBM, or others.</p> <p>Generally, you should specify a provider only if your HSM device is not supported by the default provider.</p> <p>You can configure a different provider to support keystore types other than the default. Integration Server supports both PKCS12 and JKS for keystores, but only supports JKS for truststores.</p>
Location	<p>Path location of the keystore file on the server.</p> <p>You can specify the full-path name, or a relative path in relation to the Integration Server.</p>

For this setting	Specify
Password / Re-type Password	<p>Password for the saved keystore file associated with this alias.</p> <p>If the keystore requires a password, the password must have been defined at keystore creation time using a keystore utility. Once you create the keystore alias, the keystore password is automatically saved as an Integration Server outbound password.</p> <p>Make sure you have the keystore password available when managing its corresponding keystore alias. If the keystore does not require a password, leave the fields empty.</p>
HSM-based Keystore	<p>Indicates whether the keystore file is stored on a Hardware Security Module (HSM) device. Only nCipher hardware card modules are currently supported.</p> <p>If you select this option, no path is specified in the Location field.</p>

5. Click **Submit**.
6. Enter the **Key Aliases** settings as follows:

For this setting	Specify
Password / Re-type Password	<p>Password for each alias found in the keystore.</p> <p>Most aliases require a password. If Integration Server needs to use this alias for any reason, you must provide its password.</p>
Null	<p>Indicates that no password is required for the alias.</p> <p>Select this for an alias in the keystore that is not secured with a password.</p>

7. Click **Save Changes**.

Creating Truststore Aliases

The following procedures shows how to assign aliases to truststore files.

➤ To create an alias for a truststore file

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Keystore**.
3. Click **Create Truststore Alias**.

4. Enter the **Truststore Properties** settings as follows:

For this setting	Specify
Alias	<p>A text identifier for the truststore file.</p> <p>The truststore contains the trusted CA certificates for an Integration Server, partner application, or Integration Server component.</p>
Description	Optional. A text description for the truststore alias.
Type	<p>The certificate file format of the truststore, which by default is JKS.</p> <p>Other truststore types can be made available by:</p> <ul style="list-style-type: none"> ■ Loading additional security providers. ■ Setting the <code>watt.security.trustStore.supportedTypes</code> server configuration property.
Provider	<p>The provider that is used for the truststore type. The default provider is the one shipped with the JVM, which can be Oracle, IBM, or others.</p> <p>Specify a provider only if your HSM device is not supported by the default provider.</p> <p>You can configure a different provider to support keystore types other than the default (JKS); however, Software AG does not provide support for their use.</p>
Location	<p>Path location of the truststore file on the server.</p> <p>You can specify the full-path name, or a relative path in relation to the Integration Server.</p>
Password / Re-type Password	<p>Supplied password that is used to protect the contents of the truststore.</p> <p>This password must have been defined at truststore creation time using a keystore utility. Once you create the truststore alias, its password is automatically saved as an Integration Server outbound password.</p> <p>Make sure you have the truststore password available when managing its corresponding truststore alias.</p>

5. Click **Save Changes**.

27 Controlling Access to Resources

■ Overview	486
■ Controlling Access to Resources by Port	487
■ Controlling the Use of Directives	499
■ Controlling Access to Resources with ACLs	501
■ Adding Services to a Blacklist	512

Overview

When the server receives a client's request to access a service, the server performs a number of checks to make sure the client is allowed to access the service. The server performs the following checks, in the order shown below. The client must pass all checks to access the service:

1. Does the port allow connections from this client's IP address?

The server checks allow/deny lists of IP addresses that are allowed to connect to the server through this port. If the port is an Enterprise Gateway external port and the server is licensed for webMethods Enterprise Gateway, the server also checks the Enterprise Gateway deny list. If the IP address is allowed, the server performs the next test. Otherwise, the server rejects the request.

2. Is the requested service available from this port?

The server checks allow/deny lists of services that the server makes available for execution from this port. If the service is available from this port, the server performs the next test. Otherwise the server rejects the request. The server performs this test for requests to *execute* services. It does not perform this test for requests for list, read, or write access to services.

3. Is the requested service blacklisted?

The server checks the service blacklist which identifies services for which invocation is blocked for all users. If the service is on the blacklist, the server denies access to the service and rejects the request. If the service is not on the blacklist, the server performs the next test.

4. Is the requesting user allowed to access this service?

The server checks the user name associated with the request against the appropriate access control list (ACL) associated with the service.

The server checks the user name against the List, Read, Write, or Execute ACL associated with the service. If the user belongs to a group that is listed in the ACL, the server accepts the request. Otherwise the server rejects the request.

You can configure these settings using the Integration Server Administrator.

- To limit IP addresses that connect to a port see [“Restricting IP Addresses that Can Connect to a Port” on page 487](#) below.
- To limit the services available from a port see [“Restricting the Services or Web Service Descriptors Available from a Port” on page 494](#).
- To add services to a blacklist, see [“Adding Services to a Blacklist” on page 512](#)
- To use access control lists to control which users can access an element see [“Controlling Access to Resources with ACLs” on page 501](#).

Controlling Access to Resources by Port

By default, the Integration Server provides an HTTP port at 5555 that allows all hosts (identified by their IP addresses) to connect to it and allows access to all services through that port (unless prohibited by an ACL). Although this port is ideal for initial Integration Server installation and configuration, as well as many development environments, for deployment, you should replace this port with ports that allow connections from only specified IP addresses (those of your partners and users) and make only specified services available.

Note:

By default, the Integration Server also provides a diagnostic port at 9999 that allows all hosts to connect to the server. However, users can access only the services defined with the Administrators ACL.

This section describes controlling access to resources at the port level. To control access using Access Control Lists, see [“Controlling Access to Resources with ACLs” on page 501](#).

Restricting IP Addresses that Can Connect to a Port

For any given port, you can specify IP access one of two ways:

- **Deny by Default.** Set up the port to *deny* requests from all hosts except for ones you explicitly allow. Use this approach if you want to deny most hosts and allow a few.
- **Allow by Default.** Set up the port to *allow* requests from all hosts except for ones you explicitly deny. Use this approach if you want to allow most hosts and deny a few.

You can specify these settings **globally** (for all ports) or **individually** (for one port).

Note:

If the port is an Enterprise Gateway external port and the server is licensed for webMethods Enterprise Gateway, an IP address can be further restricted by Enterprise Gateway rules. When a request from an IP address violates a rule, the server adds the IP address to a deny list that Enterprise Gateway maintains.

The following table shows where to find information about assigning the different types of IP access:

Type of access	Where to look for instructions
Controlling IP Access Globally	
Deny by Default	“Allow Inbound Connections from Specified Hosts (Deny All Others)” on page 488
Allow by Default	“Deny Inbound Connections from Specified Hosts (Allow All Others)” on page 489
Controlling IP Access of Individual Ports	

Type of access	Where to look for instructions
Deny by Default	“Allow Inbound Requests from Specified Hosts (Deny All Others)” on page 490
Allow by Default	“Deny Inbound Requests from Specified Hosts (Allow All Others)” on page 492

Note:

In Software AG Command Central, the ability to control the IP access to a port is referred to as IP Access Restrictions.

Controlling IP Access to All Ports (Globally)

This section describes how to specify the global IP access setting for ports. The server uses this setting to determine IP access for ports that do not have a custom IP access setting. The default global setting is Allow by Default.

When you create a port, you can customize IP access for it, or you can specify that it use the global IP access setting for the server. If you use the global IP access setting and later change it, the server will use the new global setting for the port. For example, as shipped, the server uses Allow by Default as the global IP access setting (with no hosts explicitly denied). If you create a new port 6666 and do not customize IP access for it, the server uses Allow by Default for port 6666. If you later change the global IP access to Deny by Default, the server will then use Deny by Default for port 6666. If you later customize IP access to port 6666, subsequent changes to the global setting will have no effect on port 6666.

To customize IP access for individual ports, see [“Allow Inbound Requests from Specified Hosts \(Deny All Others\)” on page 490](#) and [“Deny Inbound Requests from Specified Hosts \(Allow All Others\)” on page 492](#).

Allow Inbound Connections from Specified Hosts (Deny All Others)

The following procedure describes how to change the global IP access setting to Deny by Default and specify some hosts to allow.

With this setting in effect, the server denies most hosts and allows some.

Important:

Before you switch your global setting to Deny by Default, make sure you have at least one port that does not rely on the global setting and allows at least one host. If you inadvertently lock all hosts out of the server, you can correct the problem by manually updating the appropriate configuration file. See [“If You Inadvertently Deny IP Access to All Hosts” on page 492](#) for instructions.

➤ To allow inbound requests from only specified hosts

1. Open the Integration Server Administrator if it is not already open.

2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Change Global IP Access Restrictions**.
4. Click **Change IP Access Mode to Deny by Default**.

The server changes the access mode and displays a screen from which you can add hosts to the Allow List. Notice that the server has already included the host name and IP address of the machine from which you are using the Integration Server Administrator so that you are not locked out of the server.

5. Click **Add Hosts to Allow List**.
6. Specify the host names (e.g., `workstation5.webmethods.com`) or IP addresses (e.g. `132.906.19.22` or `2001:db8:85a3:8d3:1319:8a2e:370:7348`) of hosts from which the server is to accept inbound requests. Separate your entries with commas, for example: `*.allowme.com, *.allowme2.com`.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) but cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

Avoid using the fully qualified domain name of the host. Integration Server resolves incoming host names to the simple host name and then compares the simple host name to the fully qualified domain name in the allow list. The names will not match and Integration Server will conclude that the request should be denied. To work around this, you can use the * wildcard at the end of the simple host name. Alternatively, use the IP address.

Note:

IP addresses are harder to spoof, and therefore more secure.

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	<code>r*.webmethods.com</code>
?	Matches any single character	<code>workstation?.webmethods.com</code>

7. Click **Add Hosts**.

Deny Inbound Connections from Specified Hosts (Allow All Others)

The following procedure describes how to change the global IP access setting to Allow by Default and specify some hosts to deny.

With this setting in effect, the server allows most hosts and denies some.

> To deny inbound requests from specified hosts

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Change Global IP Access Restrictions**.
4. Click **Change IP Access Mode to Allow by Default**.

The server changes the access mode and displays a screen from which you can add hosts to the Deny List.

5. Click **Add Hosts to Deny List**.
6. Specify the host names (e.g., `workstation5.webmethods.com`) or IP addresses (e.g., `132.906.19.22` or `2001:db8:85a3:8d3:1319:8a2e:370:7348`) of hosts from which the server is to deny inbound requests). Separate your entries with commas, for example: `*.denyme.com, *.denyme2.com`.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) and cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

Avoid using the fully qualified domain name of the host. Integration Server resolves incoming host names to the simple host name and then compares the simple host name to the fully qualified domain name in the deny list. The names will not match and Integration Server will conclude that the request should be allowed. To work around this, you can use the * wildcard at the end of the simple host name. Alternatively, use the IP address.

Note:

IP addresses are harder to spoof, and therefore more secure.

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	<code>r*.webmethods.com</code>
?	Matches any single character	<code>workstation?.webmethods.com</code>

7. Click **Add Hosts**.

Allow Inbound Requests from Specified Hosts (Deny All Others)

The following procedure describes how to change the IP access settings for an individual port to Allow by Default and deny some hosts.

With this setting in effect, the server denies most hosts and allows some through this port.

Important:

Before you switch your global setting to Deny by Default, make sure you have at least one port that does not rely on the global setting and allows at least one host. If you inadvertently lock all hosts out of the server, you can correct the problem by manually updating the appropriate configuration file. See [“If You Inadvertently Deny IP Access to All Hosts”](#) on page 492 for instructions.

➤ **To allow inbound requests from only specified hosts**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Locate the port in the **Port List** and click the **Allow** or **Deny** link in the **IP Access** column.
4. Click **Change IP Access Mode to Deny by Default**.

The server changes the access mode and displays a screen from which you can add hosts to the Allow List. Notice that the server has already included the host name and IP address of the machine from which you are using the Integration Server Administrator so that you are not locked out of the server.

5. Click **Add Hosts to Allow List**.
6. Specify the host names or IP addresses of clients from which the server is to accept inbound requests (e.g., workstation5.webmethods.com). Separate your entries with commas, for example: *.allowme.com, *.allowme2.com.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) and cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

Avoid using the fully qualified domain name of the host. Integration Server resolves incoming host names to the simple host name and then compares the simple host name to the fully qualified domain name in the allow list. The names will not match and Integration Server will conclude that the request should be denied. To work around this, you can use the * wildcard at the end of the simple host name. Alternatively, use the IP address.

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	r*.webmethods.com
?	Matches any single character	workstation?.webmethods.com

7. Click **Add Hosts**.

Deny Inbound Requests from Specified Hosts (Allow All Others)

The following procedure describes how to change the IP access settings for an individual port to Deny by Default and allow some hosts.

With this setting in effect, the server allows most hosts and denies some through this port.

➤ To deny inbound requests from only specified hosts

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Locate the port in the **Port List** and click the **Allow** or **Deny** link in the **IP Access** column.
4. Click **Change IP Access Mode to Allow by Default**.
5. Click **Add Hosts to Deny List**.
6. Specify the host names or IP addresses of hosts from which the server is to deny inbound requests (e.g., `workstation5.webmethods.com`). Separate your entries with commas, for example: `*.denyme.com, *.denyme2.com`.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) and cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

Avoid using the fully qualified domain name of the host. Integration Server resolves incoming host names to the simple host name and then compares the simple host name to the fully qualified domain name in the deny list. The names will not match and Integration Server will conclude that the request should be allowed. To work around this, you can use the * wildcard at the end of the simple host name. Alternatively, use the IP address.

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	<code>r*.webmethods.com</code>
?	Matches any single character	<code>workstation?.webmethods.com</code>

7. Click **Add Hosts**.

If You Inadvertently Deny IP Access to All Hosts

If you change one or more ports to the Deny by Default setting, it is possible that you will prevent all hosts from gaining access to server ports.

Example 1: There are 5 ports defined. The ports are configured to use the global IP access setting and do not have allow lists for specific hosts. If you change the global IP access setting to Deny By Default, Integration Server will not allow any hosts through any ports.

Example 2: Two ports, 5555 and 7667, are defined. IP access for each port is handled by the individual port. Neither port has an allow list. If you change port 5555 to Deny by Default, then change 7667 to Deny by Default, then Integration Server will not allow hosts to connect through either port.

Use the following procedures to correct these problems.

Resetting the Global Setting IP Access Setting

➤ To reset the global setting IP access setting

1. Shut down Integration Server.
2. Edit the server.cnf file and locate the watt.server.hostAllow parameter.
3. Update the parameter to specify the IP address of a host that you want to be able to access Integration Server, as in this example:

`watt.server.hostAllow=132.906.19.22`

or

`watt.server.hostAllow=2001:db8:85a3:8d3:1319:8a2e:370:7348`
4. Save the file and restart Integration Server.
5. Navigate to the **Security > Ports** screen, click **Change Global IP Access Restrictions** and change the setting as needed. Refer to [“Controlling IP Access to All Ports \(Globally\)” on page 488](#) for more information.

Resetting the IP Access Setting for an Individual Port

➤ To reset the IP access setting for an individual port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Under **Port List**, locate the port for which you want to reset the IP access setting and click on the port number.

4. In the **View <port type> Details** screen, identify the package that is associated with it. For example, port 5555 is typically associated with the WmRoot package.
5. Shut down Integration Server.
6. Navigate to the *Integration Server_directory* `/instances/instance_name/packages/package_name/config` directory for that package and open the `listeners.cnf` file.
7. Locate the `hostAllow` parameter and update it to specify the IP address of a host you want to allow to use this port, as in the following example:

```
<array name="hostAllow" type="value" depth="1">  
<value>132.906.19.22</value>  
</array>
```

8. Save the file and restart Integration Server.
9. Navigate to the **Security > Ports** screen and update the IP access for each port as needed. Refer to [“Allow Inbound Requests from Specified Hosts \(Deny All Others\)” on page 490](#) and [“Deny Inbound Requests from Specified Hosts \(Allow All Others\)” on page 492](#) for more information.

Restricting the Services or Web Service Descriptors Available from a Port

You can limit the services or provider web service descriptors that clients can invoke through a port by setting the access mode for the port. Integration Server provides two types of port access:

- **Deny By Default.** This is the default type for newly created ports. Use this type to deny access to all services and provider web service descriptors except those you specify in a list that is associated with the port.

You might use a Deny By Default port to restrict access so only the set of services that a single application uses are accessible through the port. Set the port to Deny By Default and specify the services for the application in the list associated with the port. Then, clients using the application can only access the specific services for the application. All ports, except 5555, are initially set to Deny By Default with a limited list of services available.

- **Allow By Default.** Select this type if you intend to allow access to all services and provider web service descriptors except those you explicitly deny in a list that is associated with the port.

When Integration Server receives a service request through a port, Integration Server verifies that the service request is allowed through that port. If the service or web service descriptor can be invoked through the port, Integration Server continues with service or web service descriptor execution. If access is denied, Integration Server returns an access denied message or status to the client.

Integration Server verifies port access for the top-level service only. Integration Server does not verify port access for any child service invoked by the top-level service. For example, suppose that `serviceA` invokes `serviceB`. Additionally, suppose that port 5678 is configured to deny by default. `serviceA` is on the allow list for the port, but `serviceB` is not. When Integration Server receives a request for `serviceA` on port 5678, Integration Server verifies that `serviceA` can be invoked through the port. Integration Server does not verify that `serviceB` can be invoked through the port.

Similarly, Integration Server verifies port access for the provider web service descriptor only. Integration Server does not verify port access for any operations or handler services in the web service descriptor.

Any service URI can be included in port access control, including service URIs that are not part of the Integration Server namespace. As a result, web services that are defined on Web Services Stack can be added to the port access control lists. However, Integration Server validates and enforces web service URIs only.

Note:

By default, Integration Server performs an access check on all web services that were deployed directly into Axis2. This might be unwanted behavior. To disable the access check, set `watt.server.portAccess.axis2` to false.

The following allow list for a port that is configured to deny by default includes two web services that are not part of the Integration Server namespace, a folder (`OrderProcessing`), a provider web service descriptor, and some internal services.

Allow List for a Port Configured to Deny by Default

Allow List	
Folders and Services	Remove
/ws/EchoWS.EchoWSHttpEndpoint	✗
/ws/msh/receive	✗
OrderProcessing	✗
service:myProvideWSD	✗
wm.server.csrfguard:getCSRFSecretToken	✗
wm.server.csrfguard:isCSRFGuardEnabled	✗
wm.server.csrfguard:replaceSpecialCharacters	✗
wm.server.tx:end	✗
wm.server.tx:execute	✗
wm.server.tx:restart	✗
wm.server.tx:start	✗
wm.server.connect	✗
wm.server.disconnect	✗
wm.server:getClusterNodes	✗
wm.server:getServerNodes	✗
wm.server:noop	✗
wm.server.ping	✗

Note:

By default, the Integration Server provides an HTTP port at 5555 that allows all service requests that come in on that port access (unless prohibited by an ACL). Although this port is ideal for initial Integration Server installation and configuration, as well as many development environments, for deployment, you should replace this port with ports that limit access to services you intend to make available to your partners and users.

Note:

Another way to control access to services through a port is to restrict access to clients that present particular client certificates. See [“Customizing Authentication Using JAAS” on page 527](#) for more information.

Note:

In Software AG Command Central, the access mode functionality in which you restrict the services and web service descriptors accessible through port is referred to as URL Access.

Allow Access to Specified Services (Deny All Others)

When a port is configured to deny by default, Integration Server denies access to most services and provider web service descriptors. Integration Server allows access to the services and provider web service descriptors that you specify.

You can enter services, folders, or provider web service descriptors one at a time. Additionally, you can specify service URIs that are not part of the Integration Server namespace. As an alternative to specifying individual services or folders, you can allow all services and provider web service descriptors associated with a specific Execute ACL. For example, to create a custom Administrator port, you can expose all services or provider web service descriptors protected by the Administrators ACL.

Important:

When performing the following procedure, do not log into the server through the port you want to change. The procedure involves temporarily denying access to all services through the port. If you log on through the port you want to change and then deny access to all services through it, you will be locked out of the server. Instead, log on through a different existing port or create a new port to log on through.

➤ To allow access to specified services, folders, and provider web service descriptors

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu in the Navigation panel, click **Ports**.
3. Locate the port in the **Port List** and click the **Allow** or **Deny** link in the **Access Mode** column.
4. Click **Set Access Mode to Deny by Default**. Integration Server changes the access mode for the port.

5. Click **Add Folders and Services to Allow List**.
6. To enter the names of services, folders, or provider web service descriptors, on the left side of the screen, under **Enter one service or folder per line**, type the fully qualified name of the service, folder or provider web service descriptor for which you want to allow access. Press ENTER after each entry.

Note:

If you specify a folder, Integration Server allows access to all of the services and provider web service descriptors in the folder.

7. To specify services or provider web service descriptors by selecting from a list of elements associated with an ACL, under **Select a set of folders and services** on the right side of the screen, do the following:

- a. In the **Select an ACL** list, select the ACL used as the execute ACL for the elements for which you want to allow access.

Integration Server Administrator displays and selects all of the elements that use the selected ACL as the execute ACL.

- b. To add all of the selected items to the allow list on the left side of the screen, click **Append Selected**.

Integration Server appends the selected entries to the existing list.

- c. If you do not want to add all of the items to the allow list, deselect the ones you do not want. To deselect multiple items, press the CTRL key while deselecting. To add the remaining items to the list of allowed services for the port, click **Append Selected**.

Integration Server appends the selected entries to the existing list.

8. Repeat the above steps until you have built the list of services, folders, and provider web service descriptors you want to make available from this port.

9. Click **Save Additions**.

10. Click **Return to Ports** to return to the **Security > Ports > Edit Access Mode** screen.

Deny Access to Specified Services (Allow All Others)

When a port is configured to allow by default, Integration Server allows access to most services and provider web service descriptors. Integration Server denies access to specific services and provider web service descriptors.

You can enter services, folders, or provider web service descriptors in the deny list one at a time. Additionally, you can specify service URIs that are not part of the Integration Server namespace. As an alternative to specifying individual services or folders, you can add multiple services and

provider web service descriptors to the deny list at one time by adding all of the items associated with a specific Execute ACL.

➤ **To deny access to specified services, folders, and provider web service descriptors**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu in the Navigation panel, click **Ports**.
3. Locate the port in the **Port List** and click the **Allow** or **Deny** link in the **Access Mode** column.
4. Click **Set Access Mode to Allow by Default**. Integration Server changes the access mode for the port.
5. Click **Add Folders and Services to Deny List**.
6. To enter the names of services, folders, or provider web service descriptors, on the left side of the screen, under **Enter one service or folder per line**, type the fully qualified name of the service, folder or provider web service descriptor for which you want to deny access. Press ENTER after each entry.

Note:

If you specify a folder, Integration Server denies access to all of the services and provider web service descriptors in the folder.

7. To specify services or provider web service descriptors by selecting from a list of elements associated with an ACL, under **Select a set of folders and services** on the right side, do the following:
 - a. In the **Select an ACL** list, select the ACL used as the execute ACL for the elements for which you want to deny access.

Integration Server Administrator displays and selects all of the elements that use the selected ACL as the execute ACL.
 - b. To add all of the selected items to the deny list on the left side of the screen, click **Append Selected**.

Integration Server appends the selected entries to the existing list.
 - c. If you do not want to add all of the items to the deny list, deselect the ones you do not want. To deselect multiple items, press the CTRL key while deselecting. To add the remaining items to the list of denied services for the port, click **Append Selected**.

f appends the selected entries to the existing list.
8. Repeat the above steps until you have built the list of services, folders, and provider web service descriptors for which you want to deny access on this port.

9. Click **Save Additions**.
10. Click **Return to Ports** to return to the **Security > Ports > Edit Access Mode** screen.

Resetting a Port to the Default Access

When you reset a port to the default access, Integration Server changes IP access to the port so that only standard services that are required to connect to and authenticate to the server are available. Access to other services is denied.

> To reset a port to the default

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu in the Navigation panel, click **Ports**.
3. Locate the port with which you want to work in the **Port List** and click **Edit** in the **Access Mode** column.
4. Click **Reset to Default Access Settings**.

Integration Server changes the type to Deny By Default and creates a default list of allowed services. These include the standard services required to connect to and authenticate to the server.

Controlling the Use of Directives

A directive is a way to access or invoke resources. Integration Server supports these directives:

Directive	Used to...
admin	Route requests to the handler for the Administrator API. For more information about the Administrator API, see “ Integration Server Administrator API ” on page 971
default	Route requests to the document handler the Integration Server uses to process DSP pages.
graphql	Execute GraphQL operations, specifically, queries or mutations for GraphQL descriptors.
invoke	Run services.
odata	Process incoming OData service requests.
rad	Perform service invocations submitted as REST requests where the REST resources are configured using the URL template-based approach and must be exposed as REST API using the REST API descriptor. For more

Directive	Used to...
	<p>information about the approaches for configuring restv2 resources, see <i>REST Developer's Guide</i>.</p> <p>Note: It is recommended to use <code>rad</code> directive for the restv2 resources.</p>
<code>rest</code>	Perform service invocations submitted as REST requests where the REST resources are configured using the legacy approach. For more information about the approaches for configuring REST resources, see <i>REST Developer's Guide</i> .
<code>restv2</code>	Perform service invocations submitted as REST requests where the REST resources are configured using the URL template-based approach. For more information about the approaches for configuring restv2 resources, see <i>REST Developer's Guide</i> .
<code>soap</code>	Route requests to the Integration Server SOAP message handler. <p>Note: The <code>soap</code> directive is deprecated.</p> <p>The SOAP message handler routes messages to SOAP processors provided in versions of Integration Server prior to version 7.1.2. These processors have since been deprecated. The SOAP message handler also provides support for web services developed in Integration Server version 6.5.</p>
<code>web</code>	Access JSP files.
<code>ws</code>	Invoke web services, specifically, operations in provider web service descriptors.

Users specify directives as follows:

```
http://host:port/directive/interface/service_name
```

For example:

```
http://localhost:5555/invoke/wm.server/ping
```

By default, all Integration Server ports except the proxy port allow all the directives listed above. The proxy port allows all directives except the `web` directive. However, for security reasons, an organization typically allows only those directives that are necessary to fulfill its business requirements. You might allow all directives on ports that are accessible only to users within your firewall, but you might want to restrict directives on ports that are exposed to users outside the firewall. For example, if you want to receive only SOAP requests on a particular port, from both internal and external users, you could allow the `soap` directive but no other directives on that port. To restrict the use of directives to certain ports only, you set the `watt.server.allowDirective` parameter (see [“watt.server.” on page 1053](#)).

You can specify alternative names for the `invoke`, `soap`, and `rest` directives. For example, by default, the `invoke` directive is specified on URLs as "invoke" (that is, `http://host:port/invoke/folder/service_name`). You can identify an alternative word for users to specify as the `invoke` directive. For example, you might want to allow users to specify the `invoke` directive as "submit" (that is, `http://host:port/submit/folder/service_name`).

To specify an alternative word for the `invoke` directive, set the `watt.server.invokeDirective` parameter (see [“Server Configuration Parameters” on page 1017](#)).

To specify an alternative word for the `soap` directive, set the `watt.server.SOAP.directive` parameter (see [“Server Configuration Parameters” on page 1017](#)).

To specify an alternative word for the `rest` directive, set the `watt.server.RESTDirective` parameter (see [“Server Configuration Parameters” on page 1017](#)).

To specify an alternative word for the `restv2` directive, set the `watt.server.RESTDirective.V2` parameter (see [“Server Configuration Parameters” on page 1017](#)).

Controlling Access to Resources with ACLs

You can use Access Control Lists (ACLs) to control access to packages, folders, files, services and other elements that reside on the Integration Server. Specifically, you can control access to:

- **Services clients can invoke.** You can control which groups (and therefore which users) can invoke a service. In addition to checking ACLs to determine whether a client can invoke a service, the server performs a number of port level checks. See [“Controlling Access to Resources by Port” on page 487](#) for a description of these checks and how you can configure the server to perform them.
- **Special tools such as the Integration Server Administrator, Designer, and replicator functions.** These special abilities are granted by the Administrator, Developer, and Replicator ACLs that are provided with the Integration Server.
- **Elements that developers can see and use.** You can fine tune control over which developers have access to which packages, folders, and other elements. For example, one development group might have access to create, update, and maintain one set of services, while another development group has access to a different set. ACLs can prevent one development group from accidentally updating or damaging the work of another group.
- **Files the server can serve.** The server can serve files (for example DSP and .htm files) that reside in the `pub` directory for a package or a subdirectory of the `pub` directory. You can control access to these files by assigning ACLs to them in `.access` files. See [“Assigning ACLs to Files the Server Can Serve” on page 510](#) for more information about making files available.

This section describes how to control access to resources using ACLs. To control access at the port level, see [“Controlling Access to Resources by Port” on page 487](#).

About ACLs

ACLs control access to packages, folders, and other elements (such as services, document types, and specifications) at the group level. An ACL identifies groups that are allowed to access an

element (Allowed Groups) and/or groups that are not allowed to access an element (Denied Groups). When identifying Allowed Groups and Denied Groups, you select from groups that you have previously defined.

There are four different kinds of access: List, Read, Write, and Execute.

- **List.** Allows a user to see that an element exists. The element will be displayed on screens in Designer and Integration Server Administrator. List access also allows you to view an element's metadata.
- **Read.** Allows a user to view the main source of an element through Designer and Integration Server Administrator.
- **Write.** Allows a user to edit an element. This access also allows a user to delete or lock an element or to assign an ACL to it.
- **Execute.** Allows a user to execute a service. This access also gives the user access to files the server serves, such as DSP and .htm files.

List, Read, and Write ACLs are used mostly during development time by developers, and to some extent server administrators, who need access to create, edit, and maintain services and other elements. Execute access is used extensively in production environments.

When a user tries to access an element, the server checks the appropriate ACL (List, Read, Write, or Execute) associated with the element.

You cannot assign an ACL to an element unless you are a member of that ACL. For example, if you want to allow DevTeam1 to update the OrderForm service, you must be a member of the DevTeam1 ACL. In other words, your user name must be a member of a group that is listed in the DevTeam1 ACL. Similarly, when you change an ACL assignment for an element, you must be a member of the existing ACL and a member of the ACL to which you are assigning the element.

The following table summarizes what the different access types mean for the different elements.

Type of access and allowed actions				
Element	List	Read	Write	Execute
Package	See that the package exists. To see what the package contains, you must have List access to the elements themselves. This access is <i>not</i> inherited by other elements in the package.	N/A	N/A	N/A

Type of access and allowed actions				
Element	List	Read	Write	Execute
Folder	See that the folder exists. Children will inherit List access if they do not have a specific access of their own.	Has no meaning for the folder itself. Children will inherit Read access if they do not have a specific access of their own.	Add an element to or delete an element from the folder. Change the ACL assignment for the folder. Children will inherit Write access if they do not have a specific access of their own.	Has no meaning for the folder itself. Children will inherit Execute access if they do not have a specific access of their own.
Services (includes Flow, Java, C, XSLT, Adapter services, and web service descriptor)	See that the service exists. In Designer, the service will be listed along with non-source information.	See the service's source in the Designer.	Edit, lock, unlock, and delete the service. Change the ACL assignment for the service.	Execute the service.
Specifications, Schemas, Flat File Schemas, Document Types, Adapter Notifications, Triggers	See that the element exists.	See that the element exists. For a trigger, see the defined conditions.	Edit, lock, unlock, and delete the element. Change the ACL assignment for the element.	N/A

Package Replication

For package replication, the publishing server makes sure that the user performing the replication has replication access; that is, the user is a member of the Replicator ACL.

In addition, the publishing user must have List access to the package to see it from the publishing screens of the Integration Server Administrator. This List ACL "travels with" the package to the subscribing server. ACLs do not travel with other namespace elements, such as folders, services, etc.

On the subscribing server, the user installing the package must have List access to see it from the **Install Inbound Releases** screen. This means that the ACL must exist on the subscribing server and the installing user must be a member of that ACL. The installing user does not need Write access to the package.

Implicit and Explicit Protection

If the element is explicitly protected by an ACL, the server checks the designated ACL.

If the element is *not* explicitly protected by an ACL, the following happens:

- For elements (other than files), if the parent folder is protected by an ACL, the element inherits the folder's protection. If the folder has no explicit protection, the element inherits the protection of the folder's parent.
- For files, if the parent folder is protected by an ACL, the file inherits the folder's protection. However, if the file resides in a subfolder that is not explicitly protected by an ACL, the server assigns the Default ACL to the file. For more information about files, refer to [“Assigning ACLs to Files the Server Can Serve” on page 510](#).

Like-named folders in different packages share the same ACL. For example, if both the Finance and Marketing packages contain a top-level folder named MonthEnd, both versions of the folder are controlled by the same ACL, even though the folders have different contents.

Note:

Top-level folders never inherit List access from the parent package.

Users that Belong to More than One Group

A user can be a member of one or more groups. The following table summarizes how the server handles access for a user that is a member of a single group. Access can be any of List, Read, Write, or Execute.

If a user is a member of a group that is:	Access to the package, folder, or other element is:
Allowed	Allowed
Denied	Denied
Not-specified	Denied

But what happens when the user is a member of more than one group? To determine the access setting for a user that is member of more than one group, Integration Server assigns different strengths to the settings. Specifically, Denied overrides Allowed, and Allowed overrides Not-specified.

For example, suppose user Smith is a member of three groups, and each group has a different List access to the FY2013 package, as shown below.

Group	List access to FY2013 package
Accounting	Not-specified
Support	Denied
Corporate	Allowed

As a result of these settings, Smith is denied List access to the package because the Denied setting of the Support group overrides the other settings.

Predefined ACLs

The server comes with the following predefined ACLs. You cannot delete these ACLs.

- **Administrators.** Allows only users in the Administrators group and users with the My webMethods Administrators role access to a package, folder, or other element and denies all other users.
- **Anonymous.** Provides access to unauthenticated users (those that did not supply a userid) and users with the My webMethods Users role.
- **Default.** Allows all authenticated users and users with the My webMethods Users role access to a package, folder, or other element. When an element is not specifically assigned an ACL or does not inherit an ACL from containing folders, the server uses the Default ACL. If the ACL assigned to an element is deleted, the server uses the Default ACL. The Default ACL authorizes authenticated users only. Unauthenticated users (those that did not specify a valid userid) are authorized by the Anonymous ACL.
- **Developers.** Allows only users in the Developers group and users with the My webMethods Administrators role access to a package, folder, or other element and denies all other users.
- **Internal.** Allows only users in the Administrators and Developers groups and users with the My webMethods Administrators role access to a package, folder, or other element and denies all other users. The server assigns this ACL to built-in utility services shipped with the server, such as those in the WmRoot and WmPublic packages. You should never need to assign this ACL to an element.
- **Replicators.** Allows the Replicator user and users with the My webMethods Administrators role replication privileges.

Note:

You might see an ACL that is specific for an adapter, for example the `wmPartnerUsers` ACL. Refer to the documentation for the specific adapter for more information about its ACL.

When Does the Server Perform ACL Checking?

The Integration Server checks ACLs when:

- A client or a DSP invokes a service that resides on the Integration Server. A client can be a browser user, another Integration Server, an IS client (using the IS client API), or a custom HTTP client.
- You are using Designer to access (list, create, update, see the source of, delete, or change the ACL assignments of) an element.
- You are using the IS\X11 Administrator and you try to list or change the ACL assignment of an element.

By default, the Integration Server performs ACL checking against *externally* invoked services only. Externally invoked services are those that are directly invoked by a client or DSP. You can, however, configure a service to have its ACL checked even if it is *internally* invoked, that is, invoked by another service running on the Integration Server.

To direct the server to check the execute ACL for a service even when the service is invoked internally, use Designer to set the **Enforce Execute ACL** property to **Always**. See *webMethods Service Development Help* for more information.

Creating ACLs

When creating an ACL, you select groups to use for the Allowed Groups and Denied Groups from previously defined groups.

➤ To create an ACL

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **ACLs**.
3. Click **Add and Remove ACLs**.
4. Specify one ACL name per line. Press ENTER to separate the lines.
5. Click **Create ACLs**.

Allowing or Denying Group Access to ACLs

You can edit a new or predefined ACL to allow certain groups to access this ACL and deny permissions to other groups. You can allow and deny access to internally defined groups as well as groups and roles defined externally in a central user directory or in LDAP.

➤ To allow group access to an ACL

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **ACLs**. The server displays the Access Control Lists screen.
 - Groups in the **Allowed** list are explicitly allowed to access the packages, folders, services, or other elements associated with this ACL.
 - Groups in the **Denied** list are explicitly denied access to the packages, folders, services, or other elements associated with this ACL.

3. In the **Select ACL** list under ACL Membership, select the ACL to which you want to add groups.
4. Do one of the following:
 - If you want to allow a group or role access to this ACL, under the **Allowed** list, click **Add**.
 - If you want to deny a group or role access to this ACL, under the **Denied** list, click **Add**.
5. In the dialog box that appears, in the **Provider** list, select the location from which you want to select a user group.

If an external user directory is not configured, the **Provider** list does not appear.

6. In the **Role/Group Name** list, do one of the following:
 - If you select **Local**, select the locally defined user group for which you want to allow or deny access to the ACL.
 - If you select **Central** or **LDAP**, in the **Search** field, enter search criteria for finding a role or group. Click **Go**. Select the role or group for which you want to allow or deny access to the ACL.

Note:

The value of the **Search** field takes precedence over the value of the `watt.server.ldap.groupSearchFilter` server configuration parameter. If you do not specify a value for the **Search** field, the value of the `watt.server.ldap.groupSearchFilter` parameter is used.

7. Click **Save Changes**.

Deleting ACLs

You can delete any ACL except the predefined ACLs: Anonymous, Administrators, Default, Developers, Internal, and Replicators. You can delete ACLs that are currently assigned to packages, folder, or other elements. When a client attempts to access an element that is assigned to a deleted ACL, the server denies access.

When you delete an ACL that is assigned to a package, folder, service or other element, the Integration Server retains the deleted ACL's name. As a result, when you view the element's information, the server displays the name of the deleted ACL in the associated **ACL** field; however the server treats the ACL as an empty ACL and allows access to no one.

For information about how to assign a different ACL to a package, folder, service, or other element, see [“Assigning ACLs to Folders, Services, and Other Elements” on page 509](#).

For information about how to assign a different ACL to file, that is, a DSP or .htm file that the server serves, update the associated .access file to assign a different ACL to the file. For more information about assigning ACLs to files, see [“Assigning ACLs to Files the Server Can Serve” on page 510](#).

> To delete an ACL

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **ACLs**.
3. Click **Add and Remove ACLs**.
4. In the **Remove ACLs** area of the screen, select the ACL or ACLs you want to remove.
If an ACL is set as the List, Read, Write, or Execute ACL, the words "in use" appear in parentheses after the ACL name.
5. Click **Remove ACLs**. The server issues a prompt to verify that you want to delete the ACL.
6. Click **OK** to delete the ACL.

Default Settings and Inheritance

This section describes the default settings for newly created packages, folders, and other elements and how a *folder's* ACL assignments affect the elements it contains. For example, if you create a service and don't explicitly assign any ACLs to it, what does the server use for that service's ACL assignments? In general, it works as follows:

- When you create a package, the server assigns Default for the List ACL. (Packages do not have Read, Write, or Execute ACLs). This means that any authenticated user can see that the package exists.
- When you create a top-level folder, that is, one that is not contained in another folder, the server assigns Default for the List, Read, and Write ACLs, and Internal for the Execute ACL to the folder. This means that any authenticated user can see that the folder exists. The Read and Execute ACLs have no meaning for the folder itself. They are there just for inheritance purposes. In other words, elements in the folder will inherit those settings.
- When you create a subfolder or other element (service, schema, specification, document type, trigger, and other elements) the folder or other element inherits its ACL setting from the parent folder.

This behavior is summarized in the following table:

ACL assigned by default				
Element Type	List	Read	Write	Execute
Package	Default	N/A	N/A	N/A
Top-Level Folder	Default	Default	Default	Internal
Subfolder	Inherit	Inherit	Inherit	Inherit

ACL assigned by default				
Element Type	List	Read	Write	Execute
Other Element	Inherit	Inherit	Inherit	Inherit

What Happens When You Change Existing ACL Assignments

If you assign a specific ACL to an element then later decide to remove the ACL assignment (that is, change it to Inherited), the element will inherit the ACL of the parent folder. The server displays (**inherited**) and the name of the ACL inherited from the parent folder. If you remove an ACL assignment from a top-level folder, the server uses Default. If you remove the List ACL assignment from a package, the server uses Default.

Important:

The Default ACL identifies the Everybody group as an Allowed group and Anonymous as a denied group. This means that if an element has no ACL specifically assigned to it, then all users except unauthenticated ones can access the element. To avoid inadvertent access to resources, assign an appropriate replacement for the Default ACL.

If you change a folder's ACL assignment, it can change the ACL assignments of the elements contained within. Specifically, elements whose ACL assignment is **Inherited** will change to the folder's new ACL assignment. Elements that already have a specific ACL assignment will remain unchanged.

Assigning ACLs to Folders, Services, and Other Elements

You can use the Integration Server Administrator to assign an ACL to a folder, a subfolder, or an individual service. Keep the following points in mind when assigning ACLs using the Integration Server Administrator:

- If you assign an ACL to a folder, all the children in the folder will inherit that setting unless they already have an ACL explicitly set. For more information about inheritance, see [“Default Settings and Inheritance” on page 508](#).
- You can only assign an ACL to an element if you are a member of that ACL. For example, if you want to allow DevTeam1 to update the ProcessOrder service, you must be a member of the DevTeam1 ACL. That is, your user name must be a member of a group listed in the DevTeam1 ACL.
- If an element is locked by another user or system-locked, you cannot change the ACL assigned to the element. You can only assign an ACL to an unlocked element or an element locked by you.

Note:

Use Designer to assign ACLs to packages, specifications, document types, schemas, and triggers. For more information, see *webMethods Service Development Help*.

Use the following procedure to assign a new or different ACL to a folder or service.

> To assign an ACL to a folder or service

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click **Browse Folders**.
4. If the current screen does not list the folder or service to which you want to assign an ACL, click the name of the parent folder until the server displays a screen that lists the folder or service with which you want to work.
5. Click in the appropriate **ACL** field (**List**, **Read**, **Write**, or **Execute**).

The server displays the **ACL Information** screen. Use the pull-down list to select the ACL you want to assign to the folder or service and click **Save Changes**.

Removing an ACL from a Folder or Service

> To remove an ACL from a folder or service

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click **Browse Folders**.
4. If the current screen does not list the folder or service to which you want to assign an ACL, click the name of the parent folder until the server displays a screen that lists the folder or service with which you want to work.
5. Click in the appropriate **ACL** field (**List**, **Read**, **Write**, or **Execute**).

The server displays the **ACL Information** screen.

6. Select **<Default> (inherited)** from the pull-down menu of ACL names and click **Save Changes**.

Assigning ACLs to Files the Server Can Serve

The server can serve files that reside in the pub directory for a package or a subdirectory of the pub directory. For more information about how to serve files from the Integration Server, see the *webMethods Service Development Help*.

You control access to files by placing an .access file in the directory that contains files you want to protect. You can use an operating system tool of your choice to edit the .access file.

Note:

The .access files control access to files the server serves, such as DSP and HTML files. To control access to a *service* that a DSP or HTML file calls, you must assign an ACL to the service itself. See [“Assigning ACLs to Folders, Services, and Other Elements” on page 509](#) for more information.

If the directory contains subdirectories, they will not inherit the protection, so you must provide an .access file in each directory. For each file in the directory that you want to protect, place a line in the .access file to identify the file and the ACL you want to use to protect the file.

For example, assume you have a directory that contains three files (adminpage.dsp, home.dsp, and index.htm). You want to protect the adminpage.dsp file with the Administrators ACL so that only administrators can access this file. You want to protect the home.dsp file with the Developers ACL so only developers can access this file. You also want to assign the Default ACL to the index.htm file so all users can access it. To accomplish this, you would place the following records in the .access file:

```
adminpage.dsp Administrators
home.dsp Developers
index.htm Default
```

Note:

In the above example, because you want all users to be able to access the index.htm file, you could omit the index.htm Default from the .access file. The server uses the Default ACL for files that are not identified in an .access file or all files in a directory without an .access file.

Important: Integration Server loads .access files when a package is loaded; therefore, if you want the changes you make to take effect immediately, reload the package.

Rules for Using .access Files

When making entries in .access files, observe the following rules:

- Specify the file name only, such as adminpage.dsp followed by the ACL name. If you specify a relative path, the file will not be protected. For example, suppose file home.dsp is in subdirectory docs in directory pub (pub\docs\home.dsp). If you add the following entry to the .access file on directory pub, the file will not be protected:

```
docs\home.dsp Developers
```

Instead, add the following entry to the .access file on directory pub\docs:

```
home.dsp Developers
```

- The case in which you enter the name depends on how your file system handles case. Suppose you have a file named index.dsp. If you use a case-insensitive system such as Windows, you can enter the file name in any case. Therefore Index.dsp, INDEX.DSP, and so on are all acceptable. However, if you use a case-sensitive system such as UNIX, you must enter index.dsp.

Removing ACL Protection from a File

Use the following procedure to remove ACL protection from a file.

➤ To remove ACL protection from a file

1. Shut down the server. For instructions, see [“Shutting Down the Integration Server”](#) on page 52.
2. Edit the .access file and delete the line that specifies the file whose ACL protection you want to remove.
3. Restart the server. For instructions, see [“Restarting the Integration Server”](#) on page 55.

Adding Services to a Blacklist

Integration Server provides the ability to block the invocation of services through use of a service blacklist. An attempt to invoke a blacklisted service results in an Access Denied error. This occurs whether the blacklisted service is a top-level service directly invoked by a client or trigger, a service invoked within another service, or a service invoked from Java code using `Service.doInvoke()`. When using Designer to debug a service that calls a blacklisted service, attempts to invoke the blacklisted result in an Access Denied error as well.

The service blacklist is different from using ACLs to control which users can view, edit, or execute services. The service blacklist can be used for all services, including those installed with Integration Server such as `WmPublic` and `WmRoot` services. Additionally, the service blacklist blocks execution only and blocks it for all users. The service blacklist does not hide the specified services from select users nor does it limit execution to users in a specific group. When viewing the services in a package, a blacklisted service still appears in Designer or in Integration Server Administrator.

When a blacklisted service is invoked, Integration Server writes the following to the `server.log`:

```
[ISS.0053.0002C] Access denied for user username on port portNumber -> directive/serviceName from ipAddress
```

Note: Integration Server writes the above server and error log message whenever a user is denied access, including situations where the user provides incorrect credentials or is invoking blacklisted services.

When the Integration Server facility 0007 Authorization is set to the Debug level, Integration Server writes the following message to the server log after an attempt to invoke a blacklisted service:

```
[ISS.0007.0007D] ACLManager: service execution blocked; serviceName has been blacklisted via "watt.server.service.blacklist property"
```

Integration Server provides the server configuration parameter `watt.server.service.blacklist` for creating the blacklist. You can use a comma-separated list or a file to specify the services on the blacklist and/or the interfaces whose services or on the blacklist.

For example, to place the `pub.client:ftp` service on the service blacklist, specify the following:

```
watt.server.service.blacklist=pub.client:ftp
```

To place all the services in the `pub.client.ftp` interface (folder) on the service blacklist, specify the following:

```
watt.server.service.blacklist=pub.client.ftp
```

To place the `pub.client.ftp` interface on the service blacklist, specify the following:

```
watt.server.service.blacklist=pub.client.ftp,pub.client:ftp
```

Specifying a folder (interface) puts only the services in that folder on the service blacklist. Services that reside within a subfolder of the folder are not placed on the blacklist. For example, specifying `watt.server.service.blacklist =pub.client` places only the services in `pub.client`, such as `pub.client:ftp`, `pub.client:http`, `pub.client:restClient`, `pub.client:smtp`, `pub.client:soapClient`, and `pub.client:websocket` on the blacklist; it does not place services in `pub.client.ftp`, `pub.client.ldap`, `pub.client:oauth`, or `pub.client.sftp` on the service blacklist.

When processing the service blacklist, Integration Server does not validate whether or not a service or interface actually exists on Integration Server. However, Integration Server does ensure that service name does not contain illegal characters. If the service or interface name contains illegal characters, Integration Server logs the error:

```
[ISS.0007.0010W] ACLManager: ignored blacklist entry entryName as it is not a valid service or interface name.
```

Wildcards cannot be used in the values for `watt.server.service.blacklist` or in the file that contains the service blacklist.

Note:

Do not blacklist a service critical to the functioning of Integration Server.

Using a File for the Service Blacklist

Using a file for the service blacklist can make it easier to specify the same set of blacklisted services across the servers in your organization. Keep the following information in mind when using a file for the service blacklist:

- The file must specify each service or interface for the blacklist on a different line.
- The `watt.server.service.blacklist` value must be the relative path to the file or the absolute path to the file as where:

- A relative path has the format: `file:{relativePath}`

For a relative path, Integration Server expects the file to be in the home directory of Integration Server which is defined in the `watt.server.homeDir` parameter. If the `watt.sever.homeDir` parameter is not set, Integration Server expects to find the file in the current working directory, that is the value of the Java `'user.dir'` system property.

For example:

```
watt.server.service.blacklist=file:blacklist.txt
```

```
watt.server.service.blacklist=file:custom/blacklist.txt
```

- An absolute path has the format: `file:/directoryName/filename`

For example:

```
watt.server.service.blacklist=file:/opt/blacklist.cnf
```

- Wildcards cannot be used in the values for `watt.server.service.blacklist` or in the file that contains the service blacklist.
- If you update the contents of the service blacklist file, you must restart Integration Server for changes to take effect.
- You can set the server configuration parameter to a comma-separated list, relative path to a file, or the absolute path to a file. You cannot specify a combination of these.
- If the referenced file does not exist or cannot be read when Integration Server starts, Integration Server does not blacklist any services and continues starting up. Integration Server logs a warning message about the referenced file, specifically:

```
[ISS.0007.0009W] "ACLManager: error processing "watt.server.service.blacklist" property file.  
Stack trace: stackTrace
```

- If one of the entries in the list is a syntactically invalid name of a service or an interface, Integration Server ignores the entry.

28 Authenticating Clients

■ Overview	516
■ Basic Authentication	516
■ Digest Authentication	517
■ Kerberos Authentication	517
■ Client Certificates	518
■ Using Multiple Client Certificates	523
■ Client Authentication and Access Control	525
■ Accessing Integration Server Data through My webMethods	525

Overview

This chapter explains the types of client authentication available for use with Integration Server and details the authentication information required to configure the client side. For information on configuring authentication for the server side, see [“Configuring Integration Server for Secure Communication” on page 457](#).

The chapter explains how to configure an Integration Server client for basic authentication (user name and password) and SSL authentication. It also explains how to use Integration Server with Windows authentication, and how to make Integration Server data available to My webMethods applications.

Note:

By default, Integration Server authenticates clients that use basic, bearer, SAML, or SSL authentication schemes. For all other authentication schemes, you must create a custom JAAS login module for authenticating clients. For information about writing custom JAAS login modules, see [“Writing a Custom JAAS Login Module for Integration Server ” on page 531](#).

Basic Authentication

When the server uses basic authentication, it prompts the client for a user name and password. If a user account is found for the supplied user name, the server authenticates the user name by comparing the supplied password to the password in the user account. If the password is correct, the server proceeds with the request. If the password is not correct, the server rejects the request.

If the client does not supply a user name or password, the server uses the Default user account for the client.

Client supplied a user name/password?	User Name found?	Password correct?	Request...
YES	YES	YES	proceeds
YES	YES	NO	is rejected
YES	NO	n/a	is rejected
NO	n/a	n/a	proceeds using the Default user account

Integration Server stores user names and passwords in the *authentication cache*. The authentication cache is a caching layer in Integration Server that stores the user names and passwords in hash format.

After the first successful authentication of a user name and password (whether for a local user or central user/LDAP), Integration Server stores the credentials in the authentication cache for future reference. On subsequent authentication requests, Integration Server checks to see if the credentials already exist in the authentication cache. If the credentials already exist in the authentication cache, Integration Server does not perform any additional validation of the credentials.

Note:

Once a user has changed the password and logged in successfully with the new password, Integration Server removes the old password from the authentication cache.

You control the authentication cache through the following server configuration parameters:

- **watt.server.auth.cache.enabled.** Enables and disables the authentication cache.
- **watt.server.auth.cache.timeout.** Specifies the number of milliseconds that each cache entry can remain idle before Integration Server removes it from the authentication cache.
- **watt.server.auth.cache.capacity.** Specifies the number of user name and password combinations Integration Server stores in the authentication cache.

For more information about the server configuration parameters that control the authentication cache, see [“Server Configuration Parameters” on page 1017](#). For more information on setting up user accounts, see [“Defining a User Account” on page 84](#). You can also use externally defined user accounts. For more information on how to use external directories and how basic authentication works when using external user accounts, see [“Configuring a Central User Directory or LDAP” on page 635](#).

Digest Authentication

Integration Server supports digest authentication by processing the digest authentication credentials presented in HTTP headers. Integration Server also uses digest authentication credentials passed in web service headers to authenticate web service consumers that access an Integration Server–hosted web service. Integration Server can also send digest authentication credentials while calling third-party web services.

The password digest is a hash of the nonce, the creation time, and the password. When using digest authentication, when a client attempts to access Integration Server, Integration Server requests the client to send a password digest by concatenating the password with the nonce and the creation time. Upon receiving the password digest, Integration Server verifies the data and authenticates the client. If the password digest sent by the client matches with the password digest created by the server, the server proceeds with the request. If the password digests does not match, the server rejects the request.

To use digest authentication, you must configure the user for digest authentication while creating the user in Integration Server Administrator. You must also configure ports to use password digest for authentication of client requests. A port that is configured to use password digest for authentication of client requests will process a request from a user only if the user is configured to allow password digest for authentication.

For more information about configuring a user for digest authentication, see [“Adding User Accounts” on page 86](#).

Kerberos Authentication

Integration Server supports Kerberos authentication by processing the Kerberos tickets in the HTTP headers of service requests using the Negotiate authentication scheme. Integration Server

also uses Kerberos authentication credentials passed in web service headers to authenticate web service consumers that access an Integration Server–hosted web service.

When Integration Server receives a Kerberos ticket, it contacts the KDC using the configured principal name and principal password. If the Kerberos ticket is not valid, Integration Server rejects the request.

If the Kerberos ticket is valid, Integration Server extracts the user associated with the ticket and looks for that user in the local Integration Server user store, the Central Users, or LDAP. Software AG recommends configuring the KDC as a user directory in Central Users or LDAP so Integration Server can identify and authorize the user that is part of the Kerberos ticket submitted by the client.

For complete information about Kerberos authentication, see [“Configuring Integration Server to Use Kerberos” on page 413](#). For instructions on configuring the user directory, see [“Configuring a Central User Directory or LDAP” on page 635](#).

Client Certificates

In Integration Server, a client certificate is an X.509 digital certificate that identifies a client and is used for SSL authorization.

Checklist for Using Client Certificates

Task	Notes
Configure the server to use SSL.	For more information, see “Roadmap for Configuring SSL” on page 460 .
Import the signing certificates (trusted root certificate or certificate chain) for the client.	The trusted root certificate or certificate chain should be stored in the truststore and is required to authenticate the client. For more information, see “Certificate Mapping” on page 519 .
Configure the port to request client certificates.	For more information, see “Client Certificates and Port Configuration” on page 521 .
Import client certificates and map to specific user.	Place client certificates in separate files. Place the files in a directory to which Integration Server has access For more information, see “Importing a Certificate (Client or CA Signing Certificate) and Mapping It to a User” on page 519 .

Certificate Mapping

The *certificate mapping* feature allows you to store client certificates on an Integration Server and associate each of the certificates with a user account (for example, a certificate may be used to identify the user FINANCE). When a client presents one of these certificates, Integration Server logs the client in as the user "mapped" to the certificate.

My webMethods Server also allows you to associate a certificate and a user. If central user management is configured in Integration Server, Integration Server will automatically check the My webMethods Server database for certificate mappings when it cannot locate the user in its local store. Refer to *Administering My webMethods Server* for further details.

Important:

Be careful when mapping a user to particular client certificate. Make sure that the user's authorization level is an appropriate match.

Ports and Certificate Mappings

Integration Server automatically checks for a mapped user for certificates received at HTTPS ports. For requests received at FTPS ports, Integration Server checks for mapped users if the `watt.ftpUseCertMap` server configuration property is set to true. For more information about how client authentication works for an FTPS port, see ["FTPS Ports" on page 522](#).

If you are going to configure one or more ports to require client certificates, you *must* import the client certificates you will accept and map them to the users that you want the clients to log in as.

If you do not configure any ports to require client certificates, you might want to import client certificates and map them to users so that clients presenting these certificates can automatically log on as those users.

Importing a Certificate (Client or CA Signing Certificate) and Mapping It to a User

You import client certificates and CA signing certificates through Integration Server Administrator to keep them on file, map them to particular user accounts, and specify how they are to be used.

Keep the following points in mind before importing and mapping certificates:

- If you intend to make an SSL connection between Integration Server and an Internet resource that will serve as a client, you also need to import a copy of the client's SSL signing certificate (CA certificate).
- Although Integration Server supports loading certificates for LDAP users, Software AG recommends using central user management and then configuring LDAP and certificates in My webMethods Server.


The steps for importing client certificates and CA signing certificates are the same, and are described below.

➤ To import a client certificate and map it to a user

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Certificates**.
3. Click **Configure Client Certificates**.
4. In the **Certificate Path** field, enter the path and file name of the file that contains the certificate you want to import.

Note:

The certificate must be located on a path that is accessible to Integration Server. That is, the certificate must be on the same machine as Integration Server.

5. In the **User** field, enter a user or click  to search for and select a user.

To search for a user in the **User Name** dialog box, do one of the following:

- To select a local user, in the **Provider** list, select **Local**. Select the local user to which you want to map the certificate.

If an external user directory is not configured, the **Provider** list does not appear.


- To select a user from an external directory (LDAP or a central user directory), in the **Provider** list, select the user directory that you want to search. In the **Search** field, enter the criteria that you want to use to find a user. Click **Go**. Select the user to which you want to map the certificate.

6. In the **Usage** list, select the purpose for which you want to import this certificate. Select from one of these options:
 - **SSL Authentication.** Use the certificate to represent the client's authentication credentials when making an SSL connection with Integration Server.
 - **Verify.** Use the certificate's public key to verify the authenticity of documents, messages, or streams originating from the client and containing a digital signature.
 - **Encrypt.** Use the certificate's public key to encrypt outgoing documents, messages, or streams from Integration Server to the client.
 - **Verify and Encrypt.** Use the same certificate both to verify the authenticity of documents, messages, or streams originating from the client and containing a digital signature, and to encrypt outgoing documents, messages, or streams from Integration Server to the client.
 - **Message Authentication.** Use the certificate to represent the client's authentication credentials when making an SSL connection with Integration Server, when using *message-level* rather than transport-level authentication (for example, with web service messages whose SOAP message headers contain SSL certificate information).
7. Click **Import Certificate**.

Changing a Certificate Mapping

You can change the user to which a certificate is mapped, and the purpose for which the certificate is used.

➤ To change a user mapped to a certificate

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Certificates**.
3. Click **Configure Client Certificates**.
4. Under **Current Certificates**, in the **Subject CN** column, click the certificate for which you want to change the mapping.
5. On the **Security > Certificates > Client Certificates > Details** screen, click **Change Mapping**.
6. In the **User** field, enter the user to which you want to map the certificate or click  to search for and select a user.
7. In the **Usage** list, select the appropriate usage.
8. Click **Save Changes**.

Client Certificates and Port Configuration

Integration Server uses client certificates only for HTTPS or FTPS requests.

If a port is configured to request or require client certificates, Integration Server requests the client's certificate during the SSL handshake process, after the client authenticates the credentials presented by Integration Server. What happens next depends on how your server is configured and whether the port is HTTPS or FTPS.

HTTPS Ports

The following table shows how the Integration Server handles client requests received at an HTTPS port when different client authentication settings are in effect. These settings are specified on the **Client Authentication** parameter when configuring or editing an HTTPS port.

Parameter	Client Certificate Supplied
Username/	The server prompts the client for a user ID and password.
Password	

Parameter	Client Certificate Supplied
Digest	<p>Integration Server uses password digest for authentication of all requests. If the client does not provide the authentication information, Integration Server returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, Integration Server verifies and validates the request.</p> <p>A port that is configured to use password digest for authentication of client requests will process a request from a user only if the user is configured to allow password digest for authentication.</p>
Request Client Certificates	<p>The server requests client certificates for all requests.</p> <p>If the client does not provide a certificate, the server prompts the client for a userid and password.</p> <p>If the client provides a certificate:</p> <ul style="list-style-type: none"> ■ The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in Integration Server. If not, the client request fails, unless central user management is configured. ■ If central user management is configured, the server checks whether the certificate is mapped to a user in the central user database. If so, the server logs the client on as that user. If not, the client request fails.
Require Client Certificates	<p>The server requires client certificates for all requests.</p> <p>The server behaves as described for Request Client Certificates, except that the client must always provide a certificate.</p>

FTPS Ports

The method Integration Server uses to handle client requests received at an FTPS port depends on the various client authentication settings:

- If `watt.ftpUseCertMap` is true, Integration Server handles client requests as follows:

	Certificate	No Certificate
Username/Password	Log in with username/password supplied at prompt.	Log in with username/password supplied at prompt.

	Certificate	No Certificate
Request Certificates	If certificate is trusted and matches a mapped user, log in as that user. If certificate is not trusted or does not match a mapped user, log in with user/password supplied at prompt.	Log in with username/password supplied at prompt.
Require Certificates	If certificate is trusted and matches a mapped user, log in as that user. Ignore user/password supplied at prompt. If certificate is not trusted or does not match mapped user, ignore user/password supplied at prompt and reject the login request.	Reject the login request.

- If `watt.ftpUseCertMap` is false, Integration Server handles client requests as follows:

	Certificate	No Certificate
Username/Password	Log in with username/password supplied at prompt.	Log in with username/password supplied at prompt.
Request Certificates	Accept certificate if it is trusted, but ignore user provided in certificate. Instead, log in with user/password supplied at prompt.	Log in with username/password supplied at prompt.
Require Certificates	Accept certificate if it is trusted, but ignore user provided in certificate. Instead, log in with user/password supplied at prompt.	Reject the login request.

Using Multiple Client Certificates

Integration Server can present a single client certificate to all servers or it can present different client certificates to different SSL servers. In addition, the Integration Server can present certificates provided for this purpose by other organizations. (Some organizations prefer to provide certificates signed by their own CAs for clients to use, rather than accept the client's certificate.) You control which certificate the Integration Server presents to an SSL server by using remote server aliases or special public services.

You can set up the Integration Server to present client certificates from multiple organizations. This involves obtaining the certificates, setting them up on the server, and using remote aliases or special public services to control which certificate is being presented.

Checklist for Presenting Multiple Client Certificates

Task	Notes
Obtain copy of each certificate you want to use	You can use an existing certificate, create one, or obtain a certificate from the SSL server with which Integration Server will communicate.
Set up a remote alias	Although not required, using a remote server alias is a convenient way of directing particular certificates to particular SSL servers.
Code your flow services	How you code your flow services depends on whether or not you have defined a remote server alias for the remote server.

Importing Certificates

You must import certificates for the SSL servers with which Integration Server will communicate so that two-way SSL authentication is possible.

Once you have imported the certificates, you can centralize them in a secure location that is accessible to Integration Server; for example, in the Integration Server config directory.

Setting Up a Remote Server Alias

Using a remote server alias is a convenient method of presenting different certificates to different SSL servers.

Begin by assigning a remote server alias to the SSL server that will be presented a specific certificate (see the instructions for [“Setting Up Aliases for Remote Integration Servers”](#) on page 115). This alias will control which certificate is presented to the remote server.

If you do not use a remote server alias, you must use public services to control the client certificate that the Integration Server will present. These services, listed below, are described in more detail in the *webMethods Integration Server Built-In Services Reference*.

Service	Description
pub.security.keystore:setKeyAndChain	Specifies the key and associated certificate chain to present.
pub.security:setKeyAndChainFromBytes	Specifies the key and associated certificate chain to present, where the information is located in byte arrays (rather than files).
pub.security:clearKeyAndChain	With a set of services, use to revert the key and certificate chain back to their defaults.

Service	Description
pub.security:clearKeyAndChainFromBytes	With a set of services, use to revert the key and certificate chain back to their defaults, where the key and certificate information is located in byte arrays (rather than files).

Coding Your Flow Services

How you code your flow services depends on whether or not you have defined a remote server alias for the SSL server you want to communicate with. If you are using a remote server alias, the alias controls which certificate is presented. With a remote server alias defined, you can use the `pub.remote:invoke` services in your flow services to run services on the remote server.

If you have not defined a remote server alias, you must code your flow services to handle switching from one certificate to another using Integration Server public services. You can do so with the same public services specified in [“Setting Up a Remote Server Alias” on page 524](#).

Client Authentication and Access Control

Client authentication works with access control. After the server determines the user name of a client, it can then determine whether the client should be granted access to the requested resource. The server uses the client's group membership to control access to the server resources.

The server authenticates when a client attempts to...	The server controls access to the requested resource by determining whether...
Invoke a service	The client is a member of a group listed among the Allowed Groups or Denied Groups in the Execute ACL that is associated with the service.
Access Integration Server Administrator	The client is a member of the Administrators Group, which indicates the client has administrator privileges.
Connect to Integration Server from Designer	The client is a member of the Developers Group, which indicates the client has developer privileges.

For more information, refer to [“Configuring Integration Server for Secure Communication” on page 457](#).

Accessing Integration Server Data through My webMethods

A My webMethods Server (MWS) application, such as webMethods Monitor or webMethods Optimize for Process, may require access to data originating from Integration Server. Before the MWS application can access this data, MWS must establish a connection with Integration Server. The connection takes place as follows:

- A login request is initiated from MWS to Integration Server.

- The login credentials of the MWS user are authenticated by Integration Server.
- An Integration Server session is established.

In this situation, the user initiating the request does not need a set of Integration Server credentials. The credentials stored in the MWS user database can be used to authenticate this request. This capability is called Single Sign-On (SSO).

Important:

For SSO to work, MWS Central User Management must already be configured. For more information, see *Administering My webMethods Server*.

The underlying mechanisms for validating the login credentials of the MWS user include the Java Authorization and Authentication Service (JAAS) and the OpenSAML 1.1 library. The OpenSAML library is used to authenticate the MWS user, by resolving a SAML artifact representing the MWS user to Integration Server.

Configuring the MWS Single Sign-On Resource Setting

Although the workings of JAAS and the OpenSAML library in this context are transparent, you need to configure MWS settings to support SSO:

- Any MWS portlet that requires Integration Server data must have its Authentication Method set to hybrid. For information about configuring this setting, see *Administering My webMethods Server*.
- The MWS Single Sign-On Resource setting must be configured to the correct URL.

➤ To configure the MWS Single Sign-On Resource setting

1. In Integration Server, go to **Settings > Resources**.
2. Under Single Sign On with My webMethods Server, ensure that the following value is entered for the MWS SAML Resolver URL setting:

```
https://mws-host:mws-port/services/SAML
```

29 Customizing Authentication Using JAAS

■ Overview	528
■ Using JAAS with Integration Server	528
■ JAAS Configuration File	528
■ Pluggable Authentication Modules (PAMs)	530
■ Writing a Custom JAAS Login Module for Integration Server	531
■ JAAS Custom Login Module Example	532

Overview

This chapter explains how to set up a custom JAAS login module for use with Integration Server authentication.

Using JAAS with Integration Server

The Java Authorization and Authentication Service (JAAS) provides a standards-based mechanism for deploying custom login modules. Using JAAS, you can write your own custom login module to take over the Integration Server authentication process.

By making use of the JAAS framework for extending Java code-based security, you can customize Integration Server authentication so that multiple login modules can be called during the authentication process. JAAS allows you to specify:

- The order in which custom login modules are called.
- Whether a login module is required or optional.
- The points at which control can pass from a login module back to the controlling application.

When implementing custom login modules using JAAS, you must:

- Write the login module.
- Configure your login module within the appropriate login context in the JAAS configuration file.
- Add the JAR file containing the login module to the Integration Server classpath.

Note:

A JAAS custom login module deals only with *authentication* of Integration Server users. You cannot use the JAAS feature for Integration Server *authorization*; Integration Server uses ACLs for authorization. For more information, see [“Configuring Integration Server for Secure Communication”](#) on page 457.

JAAS Configuration File

The JAAS configuration file controls which login modules to use within a JVM. Integration Server configures the JVM to use `Integration Server_directory \instances\instance_name\config\is_jaas.cnf` as the JAAS configuration file.

A set of JAAS login modules are grouped into what is termed a *login context*. Within each login context, the login modules are specified with their full name, optional parameters, and a designation of the actions to take based on their success or failure. These designations are classified as REQUIRED, REQUISITE, SUFFICIENT, and OPTIONAL. For the login to succeed, the complete login context must succeed.

The JAAS configuration file lists the:

- Available login contexts.

- Login modules that will execute.
- Order in which the modules will execute.
- Settings that determine which actions to take if a module fails.

Following is a portion of the default JAAS configuration file for Integration Server. It shows the IS_Transport and WSS_Message_IS login contexts. The JAAS custom login modules for Integration Server include:

- *Transport-level authentication*, which is specified in the IS_Transport login context (shaded gray in the code portion below).
- *Message-level authentication* for web services, which is specified in the WSS_Message_IS login context. Integration Server message-level authentication is described in the *Web Services Developer's Guide*.

Note:

The JAAS configuration file contains additional login contexts; only IS_Transport and WSS_Message_IS (shown in the following code segments from is_jaas.cnf) are discussed here.

```
IS_Transport { /* com.wm.app.b2b.server.auth.jaas.X509ValidatorModule requisite; */
com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;
com.wm.app.b2b.server.auth.jaas.SamlOSGiLoginModule requisite; /* * The DefaultLoginModule
contains logic that provide special * default handling for Software AG products so please
leave * this module as the last module of this login context. */
com.wm.app.b2b.server.auth.jaas.DefaultLoginModule requisite;};

WSS_Message_IS { /* * Please do not rearrange the following SoftwareAG * login modules;
add your login modules before or after * these three modules */
com.wm.app.b2b.server.auth.jaas.SamlAssertLoginModule requisite;
com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;};
```

Pre-installed Login Modules

IS_Transport login context contains default login modules such as X509LoginModule, BasicLoginModule, SamlOSGiLoginModule.

Each login module first checks whether it has the required credentials. If the module finds the credentials, it uses them. If the module does not find the credentials, it returns a value of "false."

For example, the BasicLoginModule attempts to get the user name and password. If it cannot locate either, it returns a value of "false," which causes this module to fail. The modules listed within IS_Transport contain the "requisite" designation. That designation means that for the overall login to succeed, one of the modules in the login context must succeed.

X509ValidatorModule

In the above IS_Transport login context, the X509ValidatorModule is commented out because it is optional functionality. When enabled, this module performs path validation on the given X509Certificate chain, and you can configure it to perform certificate revocation lists (CRLs) checking by adding the check_crl_status and crl_url parameters as shown below:

```
IS_Transport
{
  com.wm.app.b2b.server.auth.jaas.X509ValidatorModule requisite
  check_crl_status=true
  crl_url="file:///C:\\webMethods\\sec\\crl\\lh.crl";
  com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
  com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;

  com.wm.app.b2b.server.auth.jaas.SamlOSGiLoginModule requisite;
```

The crl_url can point to a file url as shown above or it can point to a live CRL url (for example, <http://myca.com/crl/lh.crl>).

For certificate path validation, this module tries to use the truststore associated with the port on which the request came in. If the port does not identify a truststore, then this module uses Integration Server's default outbound truststore. You can override this and provide your own Integration Server truststore alias using the "trustStore_alias" property of the module as shown below:

```
com.wm.app.b2b.server.auth.jaas.X509ValidatorModule requisite
trustStore_alias="my trustStore";
```

Pluggable Authentication Modules (PAMs)

Implementing JAAS custom login modules supersedes the pluggable authentication module (PAM) approach to customizing authentication used in previous versions of Integration Server. Beginning with Integration Server version 8.0, customizing authentication with PAMs is deprecated.

PAMs will continue to work in Integration Server version 8.0 and later as an extension to the JAAS login modules BasicLoginModule and X509LoginModule:

- If you registered a PAM for the "basic" authentication mechanism, it will be treated as an extension to BasicLoginModule.
- If you registered a PAM for the "X509" authentication mechanism, it will be treated as an extension to X509LoginModule.

If a PAM is registered and the corresponding JAAS login module (BasicLoginModule or X509LoginModule) is called, authentication is first attempted on the credentials supplied for that module. If that authentication is not successful, then the PAM is called and another attempt to authenticate the user is made.

Important:

Removing the line of code specifying the `BasicLoginModule` or `X509LoginModule` in the JAAS configuration file will disable any registered PAM for the corresponding authentication mechanism.

Note:

Information about customizing Integration Server client authentication with PAMs, is available in *webMethods Integration Server Administrator's Guide version 8.0*.

Writing a Custom JAAS Login Module for Integration Server

The following sections describe in general terms how to write and deploy a custom JAAS login module for Integration Server.

Extend `SagAbstractLoginModule`

- Extend the `com.softwareag.security.jaas.login.SagAbstractLoginModule`
- Implement the following abstract methods:

```
initConfiguration()
authenticate(com.softwareag.security.jaas.login.SagCredentials
sagcredentials)
```

Integration Server collects the credentials and populates them in the `com.softwareag.security.jaas.login.SagCredentials` object. Integration Server then starts the JAAS login on the appropriate login context (for example, `ISTransport`). The JAAS framework looks in the configuration for the login modules associated with the login context. Each login module is called in turn for initialization and login; this is handled by the `SagAbstractLoginModule`, which delegates login module specific initialization and login to the derived class. For example, when JAAS login is invoked in your module, `SagAbstractLoginModule` invokes `authenticate(sagcredentials)` on your login module by passing in the `com.softwareag.security.jaas.login.SagCredentials`.

Your login module's `authenticate` method should retrieve the credentials from the `SagCredentials` object and perform the authentication (for more information, see the Javadoc for the `SagCredentials` and `SagAbstractLoginModule`).

If authentication is successful, the module populates `SagCredentials` with the correct username. If the current user name in the `SagCredentials` is not the one to be used, use the `sagcredentials.setUserName(string)` method to update the user name.

Implement `Commit()`

Depending on how the login context is configured, it is possible for more than one module to succeed with a different user name. Integration Server does not have a default mechanism to determine which course of action to take when multiple login modules succeed with different user names. To circumvent these situations, Integration Server login modules implement the following `commit` method:

```
public boolean commit() throws LoginException {
    createUserPrincipal = "true";
```

```
super.commit();  
return true;  
}
```

Here, `createUserPrincipal` is a member variable in `SagAbstractLoginModule`. The method `super.commit()` refers to the `commit()` method in `SagAbstractLoginModule`. This `commit()` method retrieves the user name from the `SagCredentials` and creates a `SagUserPrincipal` only if there are no `SagUserPrincipal` objects in the `Subject`.

Your login module should implement the `commit()` method as shown above.

If more than one login module in `IS_Transport` succeeds, only the first module that invoked `commit()` creates the principal. Thus, once you have implemented `commit()`, you can arrange the order of the login modules in the JAAS configuration file to suit your needs. If there are multiple principals in the `Subject`, Integration Server takes the principal at index 0.

If JAAS is able to authenticate the user, JAAS returns a `javax.security.auth.Subject`. Integration Server adds this JAAS subject into the current session. You can retrieve the subject by making the following call:

```
com.wm.app.b2b.server.InvokeState.getCurrentSession().getCurrentJaasSubject()
```

Place the JAR File in the Integration Server Classpath

Compile the JAR file containing your login module and place it in the Integration Server classpath. If you place the JAR file in any of your package's code \jars\static folders, Integration Server will load the file into its static classpath. After Integration Server start up, use the About page of Integration Server Administrator to check that the JAR file containing your login module appears in the Integration Server classpath.

Modify the JAAS Configuration File

Open the JAAS configuration file, *Integration Server_directory* \instances\instance_name\config\is_jaas.cnf, and add a line of code under the `IS_Transport` context that identifies your login module.

JAAS Custom Login Module Example

The following sections explain the make up of a sample JAAS custom login module with key portions of the code annotated and explained. The login module is purposely simplified, as it is designed for explanatory purposes. It is hard coded to authenticate a particular user.

JAAS Login Module for Integration Server: Sample Code

Following is the Java code for a simplified Integration Server JAAS login module. The numbers refer to code portions that are described in the summary table following the module.

```
package samples.login;
```

1.

```
import javax.security.auth.login.LoginException;
import com.softwareag.security.jaas.login.SagAbstractLoginModule;
import com.softwareag.security.jaas.login.SagCredentials;
import com.softwareag.security.jaas.principals.SagUserPrincipal;
import com.wm.app.b2b.server.UserManager;
```

2.

```
public class TestLoginModule extends SagAbstractLoginModule {
```

3.

```
    private String userId;
```

4.

```
    public boolean abort() throws LoginException {
        userId = null;
        return true;
    }
```

5.

```
    public boolean commit() throws LoginException {
        if(userId != null)
        {
            createUserPrincipal = "true";
            super.commit();
            return true;
        } else {
            return false;
        }
    }
```

6.

```
    protected void initConfiguration(){
        this.userId = null;
    }
```

7.

```
    public boolean authenticate(SagCredentials userCreds) throws
LoginException
    {
        String username = userCreds.getUserName();
        if(username == null || username.length()==0) {
            return false;
        }
        if(userCreds.getPassword() == null) {
            return false;
        }
        String password = new String(userCreds.getPassword());
        if(password == null || password.length()==0) {
            return false;
        }
    }
```

```
        if(username.equals("bob") && password.equals("123") &&
           UserManager.getUser(username) != null)
        {
            userId = username;
            return true;
        } else {
            return false;
        }
    }
    public boolean logout() throws LoginException {
        userId = null;
        return true;
    }
}
```

JAAS Custom Login Module: Code Explanation

The following table summarizes the key code portions of a custom JAAS login module in the above sample module.

Code Portion	Description
1.	Imported classes; the first is an Oracle Java class and next three are the Software AG classes from the <code>sin-common.jar</code> in <code>Software AG_directory \common\lib</code> .
2.	Identifies the JAAS custom login module as an extension of <code>SagAbstractLoginModule</code> .
3.	The user ID derived from authentication.
4.	If the login context's overall authentication fails, this method aborts the login. If the authentication attempt by the login module is successful, then this method cleans up any state information that was originally saved.
5.	This block of code shows the implementation of <code>commit</code> described in “Implement Commit()” on page 531 . The <code>commit</code> method returns <code>"true"</code> if a <code>SagUserPrincipal</code> is created by this login module, and it returns <code>"false"</code> if the login module should be ignored.
6.	Initializes the custom JAAS login module.
7.	Attempts to authenticate the user by retrieving the credentials from the given <code>com.softwareag.security.jaas.log.SagCredentials</code> object. During the <code>commit</code> phase, <code>SagAbstractLoginModule</code> creates a principal using the user name identified in the <code>SagCredentials</code> object. If the current user name in the <code>SagCredentials</code> object is not the one to be used, use the <code>sagcredentials.setUser_name(string)</code> method to update it with the correct user name. For simplification, the method specifies a hard-coded user name and password.

JAAS Configuration File: Sample Module

Following is the line of code that you would add to the IS_Transport context of the JAAS configuration file, *Integration Server_directory \instances\instance_name\config\is_jaas.cnf*, to identify the sample login module described in this section (with a requisite designation).

```
IS_Transport
{
  samples.login.TestModule requisite;
  com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
  com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;
  com.wm.app.b2b.server.auth.jaas.SamlOSGiLoginModule requisite;
  ...
};
```


30 Master Passwords and Outbound Passwords

■ Overview	538
■ Managing Outbound Passwords	538
■ Backing Up Outbound Password and Master Password Files	539
■ Changing the Master Password	540
■ Changing the Expiration Interval for the Master Password	540
■ About the configPassman.cnf File	541
■ Working with Outbound Password Settings	542
■ Working with Master Password Settings	543
■ What to Do if You Lose or Forget Your Master Password	546
■ When Problems Exist with the Master Password or Outbound Passwords at Startup ..	546
■ E-mail Listeners and Package Replication	549

Overview

As part of its normal operations, the Integration Server may connect to applications and subsystems such as remote Integration Servers, proxy servers, and databases. The Integration Server, acting as a client, is required to supply a password, referred to as an *outbound password*, to each of these systems before connecting to them. The Integration Server uses the outbound passwords to identify itself or *authenticate to* the other systems.

When you configure the Integration Server to connect to an application or subsystem, for example a database, you specify the password the Integration Server must send to the database server in order to connect to it. Later, when an Integration Server user makes a request that requires the database, the Integration Server sends the configured password to the database server and connects to it.

Note:

Whenever you create a keystore or truststore alias, the keystore or truststore password is automatically saved by Integration Server as an outbound password. For more information, see [“Using Keystores and Truststores with Integration Server” on page 477](#).

To protect these outbound passwords, the Integration Server encrypts them. By default it encrypts them using Password-Based Encryption (PBE) technology, also known as PKCS5. This encryption method requires the use of an encryption key or *master password* that you specify. The encrypted outbound passwords are stored in a file.

Note:

Flow services may also store and retrieve outbound passwords to access secure resources, using the `pub.security.outboundPasswords` services. For more information, see the *webMethods Integration Server Built-In Services Reference*.

The master password is also encrypted, and by default, is stored in a file. However, when the password is stored in a file, there is a chance that someone could access the file and decrypt the password. Therefore, for greater security, you can configure the Integration Server to prompt for the master password at server startup instead.

Note:

To protect the master password file (if you use one) and the outbound passwords file, assign them operating system Administrator access.

As stated above, outbound passwords are used by the Integration Server to authenticate to other entities. In contrast, inbound passwords are used by users and other servers to authenticate to the Integration Server. Inbound passwords are stored as a one-way hash. See [“Managing Users and Groups” on page 83](#) for a discussion of setting up inbound passwords.

The following sections describe how to manage outbound passwords.

Managing Outbound Passwords

When you first install the Integration Server, it is configured to use PBE to encrypt outbound passwords, and has a master password of "manage" with an expiration interval of 90 days.

You can change the master password and its expiration interval by using the **Security > Outbound Password** screen of the Integration Server Administrator. You can also use the Integration Server Administrator to reset the master password and all the stored outbound passwords in the unlikely event the master password or outbound passwords become lost or corrupted.

To change other settings, you must edit the `configPassman.cnf` file. Those settings are:

- Encryption method for outbound passwords
- Method the Integration Server uses to obtain the master password. The Integration Server can store the master password in a file or prompt for it at server startup.

The following table lists the tasks you can perform and where to find instructions:

To change	See
The master password.	“Changing the Master Password” on page 540
The expiration interval of the master password.	“Changing the Expiration Interval for the Master Password” on page 540
The encryption method used for outbound passwords.	“Working with Outbound Password Settings” on page 542
The location of the outbound password store.	“Working with Outbound Password Settings” on page 542
The repeat limit for the master password, that is, how soon a previously used password can be reused.	“Working with Master Password Settings” on page 543
The location of the master password store.	“Working with Master Password Settings” on page 543
All outbound passwords and the master password.	“Resetting the Master Password and Outbound Passwords” on page 548

Backing Up Outbound Password and Master Password Files

As you do with other important system files, you should regularly back up the files Integration Server uses to maintain outbound passwords and the master password. These files are located in the Integration Server instance’s home directory (*Integration Server_directory/instances/instance_name*).

The following table identifies the files that maintain outbound passwords and the master password:

File	Contents
<code>config/txnPassStore.dat</code>	Stores encrypted outbound passwords
<code>config/empw.dat</code>	Stores encrypted master password

File	Contents
config/configPassman.cnf	Specifies outbound password configuration settings
config/passman.cnf	Non-editable version of configPassman.cnf

Always back up and restore these files together. If you change the name or location of the outbound password store or the master password store, make sure your backup procedure backs up the correct files.

Changing the Master Password

When you first install the Integration Server, the master password is "manage." For security purposes, you should change the master password immediately after installation and again on a regular basis. You should also change it when there are personnel changes.

The default expiration interval for a master password is 90 days. As the expiration date nears, the Integration Server displays the password expiration status on the Integration Server and sends warning messages to the server console stating that it is time to change the master password. If the Integration Server is configured for e-mail notification, the Integration Server will also send e-mail messages with this information to the configured addresses.

Keep the following points in mind when changing the master password:

- To keep outbound passwords synchronized with the master password, the Integration Server does not process requests to store and retrieve outbound passwords while the master password is being changed. Therefore, if your system has many aliases, consider performing the master password change during off peak hours to prevent any decrease in performance.
- If you have lost your master password, refer to [“Determining Whether You Can Restore the Passwords” on page 547](#).

➤ To change the master password

1. Open the Integration Server Administrator.
2. In the Security menu of the Navigation panel, click **Outbound Passwords**.
3. Click **Update Master Password**.
4. Enter the current password, then enter and confirm the new password.
5. Click **Change Password**.

Changing the Expiration Interval for the Master Password

The default expiration interval for a master password is 90 days. You can see the current expiration date by looking at the **Security > Outbound Passwords** screen.

The expiration interval is a *recommended* time between password changes. If you do not change the master password by the expiration date, the Integration Server will continue to operate using the existing password indefinitely.

Although it is not recommended, you can specify an interval of 0. With this setting, the password will not expire and no warnings will be sent to the Integration Server Administrator or the server log.

➤ To change the expiration interval for the master password

1. Open the Integration Server Administrator if it is not already open.
2. In the Security menu of the Navigation panel, click **Outbound Passwords**.
3. Click **Update Expiration Interval**.
4. Enter the new expiration interval in days and click **Update**. The maximum interval is 366 days.

About the configPassman.cnf File

The configPassman.cnf file contains additional configuration settings for outbound password encryption that are not contained in the Integration Server Administrator. The file consists of a number of properties, some of which are commented out in the default configuration.

Important:

The configPassman.cnf file has a companion file, passman.cnf. If you make changes to configPassman.cnf file, the Integration Server will automatically update passman.cnf to reflect these changes when you initialize the Integration Server. Never update passman.cnf directly.

As shipped, the configPassman.cnf file specifies that outbound passwords will be stored in file *Integration Server_directory /instances/instance_name/config/txnPassStore.dat* and encrypted using Password-Based Encryption (PBE). In addition, it specifies that the master password will be stored in file *Integration Server_directory /instances/instance_name/config/empw.dat*. Properties that can be used to specify other settings are commented out.

If you want to change these optional settings, you must edit the configPassman.cnf file. The file must always specify the following:

- Encryption method for outbound passwords
- Location of the file that contains the outbound passwords
- Method the Integration Server uses to obtain the master password

The following sections describe the configPassman.cnf file in detail and how to change outbound password and master password settings.

Working with Outbound Password Settings

The `configPassman.cnf` file contains properties that specify the name, location, and encryption for the outbound password file.

Important:

Determine your strategy for outbound password and master password behavior *before* you launch and configure your Integration Server the first time. If you change these settings after the Integration Server has been configured, the master password and outbound passwords can become out of sync.

See [“Working with Master Password Settings” on page 543](#) for instructions on using the `configPassman.cnf` file to change master password settings.

If you are not already familiar with the `configPassman.cnf` file, read [“About the configPassman.cnf File” on page 541](#) above before proceeding.

Controlling Name and Location of Outbound Password File

The default file name and location for the outbound password file is determined by the `outbound.password.store.field.fileName` property in the `config/txnPassStore.dat` file located in the server instances home directory.

Keep the following information in mind when changing the value of the `outbound.password.store.field.fileName` property:

- The `outbound.password.store.field.fileName` property must always be present and uncommented in the `configPassman.cnf` file.
- You can specify an absolute or relative path.
- In the path name, use the forward slash (/) only; the backward slash (\) is not supported.

➤ To change the file name or location of the outbound password file

1. Use a text editor to open the following file:

Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. Locate and edit the following property: `outbound.password.store.field.fileName`
3. Save and close the `configPassman.cnf` file.

Controlling Encryption of Outbound Password File

The `configPassman.cnf` file contains properties for each of the available encryption methods for the outbound password file. The default encryption method for the outbound password file is Password-Based Encryption (PBE).

Uncomment the property for the encryption method that you want the outbound password file to use and comment out the properties for the encryption methods that you do not want to use. *One and only one* of the properties must always be uncommented.

➤ **To change the encryption method for the outbound password file:**

1. Use a text editor to open the following file:

Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. Uncomment one of the properties in the following table to indicate the encryption that you want the outbound password file to use.

Property	Description
default.encryptor=EntrustPbePlus	Password-Based Encryption, which is the most secure. This is the default.
default.encryptor=Base64	Base64 encoding, which is not secure.
default.encryptor=None	Clear text, which is not secure.

3. Comment out the remaining properties so that *one and only one* of the properties is uncommented.
4. Save and close the configPassman.cnf file.

Working with Master Password Settings

The configPassman.cnf file contains properties that you can use to change settings for the master password. By default, the master password is stored in the file config/empw.dat under the server instance's home directory. If you prefer, you can configure Integration Server to prompt for the master password at server initialization.

For more information about storing the master password in a file, see [“Storing the Master Password in a File” on page 544](#).

For more information, see [“Prompting for the Master Password at Server Initialization” on page 544](#).

If you are not already familiar with the configPassman.cnf file, read [“About the configPassman.cnf File” on page 541](#) before proceeding.

Important:

Determine your strategy for outbound password and master password behavior *before* you launch and configure your Integration Server the first time. If you change these settings after the Integration Server has been configured, the master password and outbound passwords can become out of sync.

See “[Working with Outbound Password Settings](#)” on page 542 for instructions on using the `configPassman.cnf` file to change outbound password settings.

Storing the Master Password in a File

The `configPassman.cnf` provides properties that you can use to configure the master password store.

➤ To configure the master password store for a file

1. Use a text editor to open the following file:

Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. Set the properties as described in the following table:

Property	Description
<code>master.password.storeInFile</code>	Controls whether Integration Server stores the master password in a file (true). Set to true.
<code>master.password.field.fileName</code>	Location of the master password store. The default is: <code>config/empw.dat</code> When specifying the path, use the forward slash (/) only. The backward slash (\) is not supported.
<code>master.password.field.repeatLimi</code>	Indicates the number of password changes required before you can reuse a password. The default is 3.

3. Save and close the `configPassman.cnf` file.

Prompting for the Master Password at Server Initialization

The `configpassman.cnf` file provides properties that you can use to prompt for the master password at server initialization.

Use these properties only if you want Integration Server to prompt for the password at server initialization (that is, you specified `false` for `master.password.storeInFile`).

Keep the following information in mind when deciding whether to configure Integration Server to prompt for the master password at server initialization:

- To prompt for the master password at server initialization, the `master.password.storeInFile` property must be set to `false`.

- Integration Server runs in headless mode as controlled by the following property in `custom_wrapper.conf`: `wrapper.java.additional.205=-Djava.awt.headless=true`

If you want Integration Server to prompt for the master password during server initialization, you must edit `custom_wrapper.conf` to have the following:

```
wrapper.java.additional.205=-Djava.awt.headless=false
```

For more information about editing the `custom_wrapper.conf` file, see [“Changing Settings in the Configuration File `custom_wrapper.conf`” on page 47](#).

Note:

To run Microservices Runtime in headless mode, edit the `JAVA_CUSTOM_OPTS` property in `Integration Server_directory/bin/setenv.bat(sh)` to include the following property:

`java.awt.headless=true` set for the For information about passing Java system properties to Microservices Runtime, see *Developing Microservices with webMethods Microservices Runtime*

- You cannot configure Integration Server to prompt for the master password at server initialization if:
 - Integration Server runs as a Windows service.
 - Integration Server runs as a background application on UNIX.
- If you do not want Integration Server to prompt for the password at server initialization, make sure to comment out the following properties in the `configpassman.cnf` file:


```
master.password.field.useGUI and master.password.field.attemptsLimit.
```

➤ **To configure Integration Server to prompt for master password at server initialization**

1. Use a text editor to open the following file:

```
Integration Server_directory/instances/instance_name/config/configPassman.cnf
```

2. Make sure `master.password.field.useGUI` and `master.password.field.attemptsLimit` are not commented out. If the parameters do not exist in the file, add them.
3. Set the properties as described in the following table.

Property	Description
<code>master.password.storeInFile</code>	Controls whether Integration Server stores the master password in a file. Set to false.
<code>master.password.field.usgeGUI</code>	Specifies where Integration Server prompts for the master password. Set to true to prompt for the password in a popup window. If you choose this method, you can start Integration Server from the Windows start menu.

Property	Description
	Set to false to prompt for the password on the server console. If you choose this method, you cannot start Integration Server from the Windows start menu. The default is true.
<code>master.password.field.attemptsLimit</code>	Indicates the number of unsuccessful login attempts permitted before Integration Server rejects the request. The default is 3.

4. Save and close the `configPassman.cnf` file.

What to Do if You Lose or Forget Your Master Password

If your Integration Server is configured to encrypt outbound passwords using Password-Based Encryption (PBE), your Integration Server will have a master password, which is the key used to encrypt outbound passwords. You need to enter the master password whenever you want to change to a new encryption key. In addition, some installations are configured so that Integration Server prompts for the master password when Integration Server initializes; without the password, Integration Server will start up in *safe mode*, which is described below.

Therefore, if you lose or forget your master password, you will need to restore it or reset it, depending on the circumstances. For more information, see [“Determining Whether You Can Restore the Passwords”](#) on page 547.

When Problems Exist with the Master Password or Outbound Passwords at Startup

If Integration Server detects a problem with the master password or outbound passwords at startup, the server places you in *safe mode*, which is a special mode from which you can diagnose and correct problems.

When Integration Server is in safe mode, the server displays the Integration Server Administrator, but Integration Server is not connected to any external resources.

When you are placed into safe mode because of problems with the master password or outbound passwords, you will see the following message in the upper left corner of the Server Statistics screen of the Integration Server Administrator:

```
SERVER IS RUNNING IN SAFE MODE. Master password sanity check failed -- invalid
master password provided.
```

Important:

When you are in safe mode, do not configure or modify outbound passwords unless they have been reset as part of the **Reset All Outbound Passwords** task.

When there is a problem with these passwords, you can correct the problem by restoring the passwords or resetting them. The method you choose depends on the problem with the passwords. There are a number of reasons the Integration Server will automatically go into safe mode.

Passwords are Corrupted or Out of Sync

It is possible that the master password file, outbound password file, or both are corrupted. It is also possible that these files are out of sync with each other. The files are out of synch when the key used to encrypt the contents of the outbound password file is not the key in the master password file. In either case, refer to [“Determining Whether You Can Restore the Passwords” on page 547](#) for instructions.

You Entered the Wrong Master Password by Mistake

You might be in safe mode because you unintentionally entered the wrong master password when prompted for it at server startup. If you think this is the case, shut down Integration Server and restart it, this time specifying the correct master password when prompted.

Platform Locale Has Changed

Any change to the OS locale or default encoding can render the outbound password and master password files unreadable by Integration Server. For this reason, Software AG recommends that you do not change platform locale after the Integration Server has been installed and started.

Determining Whether You Can Restore the Passwords

You can restore passwords if either of the following is true:

- Your master password and outbound passwords are stored in files and you have recent backups of both and the `passman.cnf` file.
- Integration Server is configured to prompt for the master password, you have a recent backup of the outbound password file and the `passman.cnf` file, and you know the master password for that backup.

You must reset the passwords if any of the following is true:

- Your master password and outbound passwords are stored in files and you do *not* have recent backups the master password file, the outbound password file, and the `passman.cnf` file.
- Integration Server is configured to prompt for the master password and you do *not* have recent backups of the outbound password file and the `passman.cnf` file.
- Integration Server is configured to prompt for the master password and you have lost or forgotten the master password.

Restoring the Master Password and Outbound Password Files

Before restoring these files, make sure you have read [“Determining Whether You Can Restore the Passwords” on page 547](#) to determine if you can restore or need to reset.

- **To restore the master password and outbound password files**

1. Determine which files you need to restore.

If your master password is *not* stored in a file, that is, your Integration Server prompts you for a master password at server startup, then you can restore just the outbound password file and the passman.cnf file. Otherwise, you must restore the master password file, the outbound password file, and the passman.cnf file from backups.

2. Determine the name and location of the files.

The passman.cnf file is always config/passman.cnf located under the server instance's home directory (*Integration Server_directory \instances\instance_name*). By default, the master password file is config/empw.dat and the outbound password file is in config/txnPassStore.dat. If you are not sure of the location of these files on your system, look at the file config/configPassman.cnf. For information about using this file, see [“About the configPassman.cnf File” on page 541](#).

3. Shut down the Integration Server.
4. Copy the replacement files to appropriate directory.
5. Restart Integration Server.

Note:

Always back up and restore the master password file (if you use one), the outbound password file, and the passman.cnf file together.

Resetting the Master Password and Outbound Passwords

Before resetting these passwords, make sure you have read [“Determining Whether You Can Restore the Passwords” on page 547](#) to determine if you really need to reset the passwords or if you can restore them instead.

The reset procedure clears (blanks out) the stored outbound passwords and resets the master password to "manage." In addition, you must manually re-enter the appropriate passwords for all application and subsystem passwords on their respective configuration screens in the Integration Server Administrator.

> To reset stored outbound passwords and the master password

1. Start Integration Server if it is not already running. If your Integration Server is configured to prompt you for a master password during server initialization, enter any value.

Integration Server takes you into *safe mode*, which is the Integration Server Administrator, but in a mode that is not connected to any external resources.

2. In the Security menu of the Navigation panel, click **Outbound Passwords**.
3. Click **Update Master Password**.

4. Click **Reset All Outbound Passwords**.

Integration Server displays a warning screen, to be sure you want to reset the passwords.

5. Click **Reset Passwords**.

The Integration Server asks again if you are sure you want to reset the passwords.

6. Click **OK**.

This step clears the stored outbound passwords and changes the master password to "manage."

7. From the **Outbound Passwords** screen, click **Change Password** and change the master password to something other than "manage."

8. Restart Integration Server.

You will see error messages as the Integration Server attempts to connect to the applications and subsystems for which the server no longer has passwords stored.

9. Go to the configuration screen for each application or subsystem and re-enter the password required for Integration Server to connect to that application or subsystem. Screens to check include those that define remote server aliases, cluster configuration, JDBC connection pools, e-mail listeners, LDAP servers, proxy servers, Broker configuration, and WmDB.

E-mail Listeners and Package Replication

When you export a package that is associated with a listener, information about that listener is sent with the package. However, in the case of an e-mail listener, not all the listener configuration information is sent to the destination Integration Server. Specifically, the outbound password that the e-mail listener uses to connect to the e-mail Server is not sent. As a result, when the listener on the destination Integration Server tries to connect to the e-mail server, the connection fails. Although the listener appears on the list of ports, it will not be enabled. You will also see error messages on the server console.

To enable the port, go to the **Security > Ports > Edit E-mail Client Configuration** Screen in the Integration Server Administrator and update the **Password** field to specify the password needed to connect to the e-mail server.

If you export a package that is associated with an e-mail listener from a 6.5 Integration Server to a pre-6.5 Integration Server, the e-mail listener will not be replicated at all. You must manually reconfigure the listener on the pre-6.5 Integration Server after installing the package there.

31 Securing Integration Server with CSRF Guard

■ What is CSRF?	552
■ How Does Integration Server Prevent CSRF Attacks?	552
■ Understanding CSRF Guard Terminology	552
■ Configuring CSRF Guard in Integration Server	554
■ Limitations when Configuring CSRF Guard in Integration Server	556

What is CSRF?

Cross-Site Request Forgery (CSRF) is one of the most common attacks on websites and web applications. A CSRF attack occurs when a user inadvertently loads a webpage that contains a malicious request. This webpage sends a malicious request to a website or web application using the identity and privileges of the user to perform an undesired action, such as changing configurations or invoking a service.

A web application is vulnerable to CSRF attacks if the application performs actions based on inputs from authenticated users but does not require users to authorize specific actions. That is, if you are authenticated to a web application by a cookie stored in your web browser, you could unknowingly send a malicious HTTP or HTTPS request to the application.

How Does Integration Server Prevent CSRF Attacks?

Integration Server uses the CSRF guard feature to prevent CSRF attacks. Integration Server prevents CSRF attacks by creating one CSRF secure token per session when it receives authorization requests from Integration Server Administrator or other client applications. Integration Server adds this CSRF secure token to subsequent requests until the session expires. The CSRF token expires when the session ends.

When you send a request, Integration Server verifies the existence and validity of the token in the request and compares it to the token in the session. If there is no token in the request, or if the token in the request does not match the token in the session, Integration Server terminates the request. Integration Server also logs the event as a potential CSRF attack in the server log and the security audit log.

You use the Integration Server Administrator to enable or disable CSRF guard in Integration Server.

Integration Server inserts and verifies CSRF secure tokens for:

- HTTP requests from a web browser for dynamic server pages (DSPs)
- HTTP request for invoke, rest, or restv2 directives
- Ajax XMLHttpRequests

Understanding CSRF Guard Terminology

Before configuring CSRF guard in Integration Server, you may find it helpful to first understand the following terminology used with reference to CSRF guard in Integration Server:

- **Excluded User Agents.** The user agent value is the string that corresponds to the `User-Agent` HTTP header in the HTTP request. Excluded user agents are user agents for which Integration Server does not enforce CSRF guard. That is, Integration Server will not check for CSRF tokens on requests from these excluded user agents.

You can specify the user agents as regular expressions. The asterisk (*) is the only wildcard character allowed in the regular expression. To separate the entries, enter one user agent per line. Press ENTER to separate the lines.

For example:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

```
Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us)
AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341
Safari/528.16
```

```
*Mozilla*
```

- **Landing Pages.** The home pages of Integration Server packages are referred to as landing pages. Integration Server will not check for CSRF secure tokens in the landing pages, but the server will insert a token for that page. Integration Server guards all further requests from these landing pages with CSRF secure tokens.

You cannot specify landing pages as regular expressions. To separate the entries, enter one landing page per line. Press ENTER to separate the lines.

For example:

```
MyPackage/index.dsp
```

```
MyPackage/index.html
```

- **Unprotected URLs.** The URLs for which Integration Server does not have to check for CSRF secure tokens are referred to as unprotected URLs. Integration Server requires that the requests coming from all URLs that are not specified in this field must contain CSRF secure tokens.

If you specify a DSP page as an unprotected URL, Integration Server will not insert a CSRF secure token for that file. If you attempt to access a protected page from this DSP page, Integration Server issues an error or redirects you to the home page of Integration Server Administrator depending on your Denial Action configuration.

You can specify unprotected URLs as regular expressions. The asterisk (*) is the only wildcard character allowed in the regular expression. To separate the entries, enter one URL per line. Press ENTER to separate the lines.

The following table provides examples of URLs for **Unprotected URLs** text area.

In this example...	Integration Server does not check for CSRF secure token in...
MyPackage/abc.dsp	The abc.dsp page in the MyPackage package.
MyPackage/*	All the pages in the MyPackage package.
invoke/pub.math:addInts	A request invoking the pub.math:addInts service.
invoke/pub*	Requests invoking all services starting with “pub”.
invoke/*	Any invoke requests.

- **Denial Action.** The action you want Integration Server to perform when it detects that a request does not contain a CSRF secure token or contains an invalid CSRF secure token. You can configure Integration Server to:
 - Redirect the user to the home page of Integration Server Administrator or to a webpage that displays a warning that Integration Server has detected a CSRF attack.
 - Issue an error and terminate the request.

Configuring CSRF Guard in Integration Server

Keep the following points in mind when you configure CSRF guard in Integration Server:

- When you enable or disable CSRF guard in Integration Server, you must refresh the web browser for the changes to take effect.
- Integration Server does not provide protection from CSRF attacks in the following situations:
 - Requests from users with execute access to invoke services that have the Anonymous ACL assigned to them.
 - Requests from Integration Server Administrator or a client application after an Integration Server session has timed out. In such cases, Integration Server will redirect the user or issue an error. You must refresh the web browser to continue.
 - Requests from user agents that are different from the user agent that was used when creating the session.

➤ To configure CSRF guard in Integration Server

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Security > CSRF Guard > Edit CSRF Guard Settings**.
3. Select the **Enabled** check box to enable CSRF guard in Integration Server.
4. In the **Excluded User Agents** text area, enter the user agents for which CSRF guard is not to be enforced.

The user agent value is the string that corresponds to the `User-Agent` HTTP header in the HTTP request.

You can specify the user agents as regular expressions. The asterisk (*) is the only wildcard character allowed in the regular expression. To separate the entries, enter one user agent per line. Press ENTER to separate the lines.

5. In the **Landing Pages** text area, enter the list of landing pages for the packages in your Integration Server. Integration Server will not check for CSRF secure tokens in the landing pages, but the server will insert a token for that page. Integration Server guards all further requests from these landing pages with CSRF secure tokens.

You cannot specify landing pages as regular expressions. To separate the entries, enter one landing page per line. Press ENTER to separate the lines.

6. In the **Unprotected URLs** text area, enter the URLs for which Integration Server does not have to check for CSRF secure tokens.

You can specify unprotected URLs as regular expressions. The asterisk (*) is the only wildcard character allowed in the regular expression. To separate the entries, enter one URL per line. Press ENTER to separate the lines.

Note:

Do not specify landing pages in the **Unprotected URLs** text area. If you specify landing page URLs in both **Landing Pages** and **Unprotected URLs** text areas, the landing page option will take precedence and Integration Server will not check for CSRF secure tokens in those pages.

7. From the **Denial Action** options, select the action that you want Integration Server to perform when it detects that a request does not contain a CSRF secure token or contains an invalid CSRF secure token.

Select...

To...

Error

Issue an “access denied” error and terminate the request. This is the default.

If you select **Error**, Integration Server issues the following error when it detects that a request does not contain a CSRF secure token or contains an invalid CSRF secure token:

```
Access Denied. Invalid CSRF secure token.
```

This error message suggests that Integration Server has detected a CSRF attack.

Integration Server also issues this error message for the following situations:

- Integration Server session has expired.
- The web browser is not refreshed after enabling CSRF guard.
- Another user who is connected to the same Integration Server restarted Integration Server.

In these cases, refresh the web browser to continue.

Redirect

Redirect the user as follows:

- If the CSRF threat is detected when a user accesses a DSP page in the Integration Server Administrator, redirect the user to the home page of Integration Server Administrator.

Select...

To...

- If the CSRF threat is detected in a URL or client request that includes an `invoke`, a `rest`, or a `restv2` directive, redirect the user to a webpage that displays a warning that Integration Server has detected a CSRF attack. The user must click **Continue** to execute the service.

Note: Integration Server redirects the user to this page only if the client application accepts `text/html` as the content type. If the client application does not accept `text/html`, Integration Server returns an access denied error.

8. Click **Save Changes** and refresh the web browser for the changes to take effect.

Limitations when Configuring CSRF Guard in Integration Server

- When you enable or disable CSRF guard in Integration Server, you must refresh the web browser.
- You cannot use the CSRF guard feature if your Integration Server runs as part of a non-clustered group of Integration Servers in which the `ISInternal` functional alias on each server points to the same database.
- Integration Server does not insert CSRF secure tokens in custom DSP pages that use JavaScript Location object, such as `document.location` and `window.location.href`. You must update these pages manually.

You do not have to define the JavaScript variables `_csrfTokenNm_`, `_csrfTokenVal_`, `is_csrf_guard_enabled`, and `needToInsertToken`. But, you must import `Integration Server_directory \instances \instance_name \packages \WmRoot \csrf-guard.js` to your DSP before using these variables.

- Integration Server inserts CSRF secure tokens in the links in DSPs only if these links point to a DSP. If these links do not point to a DSP, you must update these links manually to include the CSRF secure tokens.

For example, if you have the following code in your DSP:

```
<a href="/invoke/wm.sap.Transaction/viewAs?type=xml">
```

You must replace it with the following code:

```
<a href="/invoke/wm.sap.Transaction/viewAs?type=xml&secureCSRFToken=%value
secureCSRFToken%"></a>
```

For more information about using CSRF guard in DSPs, see *Dynamic Server Pages and Output Templates Developer's Guide*.

32 Configuring webMethods Enterprise Gateway

■ Overview	559
■ How Enterprise Gateway Works	559
■ Version Interoperability Between Enterprise Gateway Server and Internal Server	562
■ Advantages of Enterprise Gateway over Traditional Third-Party Proxy Servers	563
■ About Denial of Service Protection	564
■ About Mobile Application Protection	565
■ About Mobile Data Synchronization	565
■ About SQL Injection Protection	565
■ About Antivirus Scan Filter	566
■ About Custom Filter	568
■ Clustering in the Enterprise Gateway Configuration	568
■ Setting Up an Enterprise Gateway	569
■ Configuring the Enterprise Gateway Ports	571
■ Connecting Your Internal Server to an Enterprise Gateway Server	576
■ Viewing Connections to the Enterprise Gateway Registration Port	580
■ Performing Client Authentication on Enterprise Gateway Server	581
■ Working with Enterprise Gateway Rules	582
■ Specifying Alert Options	589
■ Preventing Denial of Service Attacks	591

- Controlling Use of Mobile Applications 594
- Frequently Asked Questions about Enterprise Gateway 596

Overview

If your Integration Server sits behind an internal firewall and is not allowed to accept communications from external clients through the DMZ, you can set up a webMethods Enterprise Gateway to allow this server to process requests from external clients such as mobile applications.

In an Enterprise Gateway configuration, the Integration Server that remains behind the inner firewall is referred to as an *Internal Server*. You place another Integration Server, called an Enterprise Gateway Server, in your DMZ. The Enterprise Gateway Server acts as an intermediary between the external clients and your Internal Servers to protect your Internal Servers and their applications, services, and data from malicious attacks. Enterprise Gateway Server supports nearly all requests that a regular Integration Server handles, including guaranteed delivery.

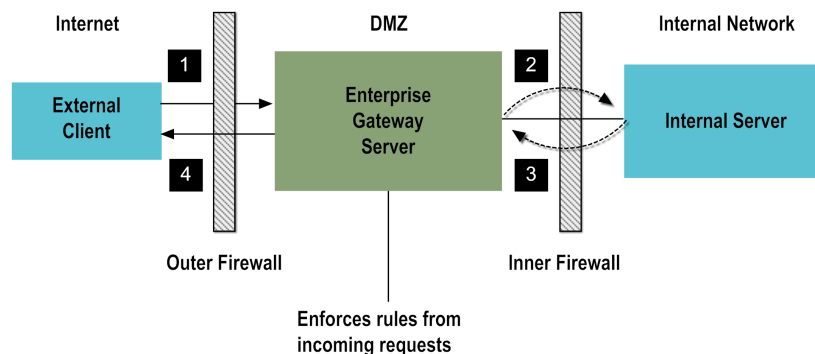
By default, all user validation and transaction processing is performed on the Internal Server, but you can configure Enterprise Gateway Server to perform user validation. You can also set up rules on Enterprise Gateway Server to filter requests before they are passed to the Internal Server.

Important:

To use Enterprise Gateway, you must have a webMethods Enterprise Gateway license. The license for Enterprise Gateway allows only partial Integration Server functionality. Install Enterprise Gateway on its own host Integration Server. Do not install other products on an Integration Server that hosts Enterprise Gateway.

How Enterprise Gateway Works

The following diagram illustrates how an external client request is handled in an Enterprise Gateway configuration:



External clients send requests to Enterprise Gateway Server (1). Enterprise Gateway Server collects client information from each request and evaluates the request against any Enterprise Gateway rules that have been defined. Enterprise Gateway Server then passes requests that have not violated a rule to the Internal Server (2). The Internal Server processes the requests and sends responses to Enterprise Gateway Server (3). Enterprise Gateway Server then passes the responses back to the client (4).

Enterprise Gateway Ports

The Integration Server that functions as an Enterprise Gateway Server uses an Enterprise Gateway *external port* to listen for requests from external clients and an Enterprise Gateway *registration port* through which it maintains its connection to the Internal Server. For security purposes, the Internal Server initiates the outbound connections to the registration port.

Note:

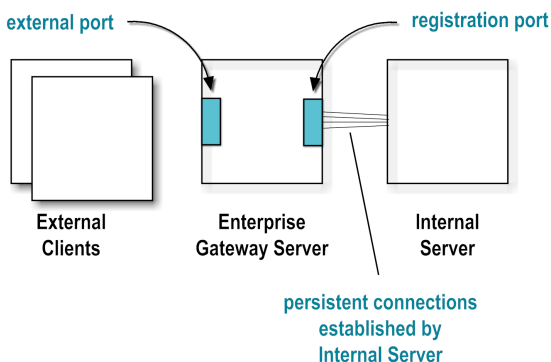
You cannot configure an Enterprise Gateway *external port* and Enterprise Gateway *registration port* if webMethods API Gateway is installed on Integration Server. In this case, Enterprise Gateway Server and Internal Server ports are disabled.

By limiting the connections to just those established by the Internal Server, the Enterprise Gateway makes it more difficult for attackers to directly penetrate your internal network, even if they subvert a system in the DMZ. However, like any other security mechanism, it is not foolproof; the information still flows from the DMZ to the internal network over the connection established from inside the firewall.

Important:

To get the maximum benefit from the Enterprise Gateway configuration, Software AG highly recommends that you configure your inner firewall to deny all inbound connections. With this configuration, you isolate the Internal Server from the DMZ. This capability is the main advantage of using an Enterprise Gateway Server over traditional third-party proxy servers.

The following diagram shows the location of the Enterprise Gateway external and registration ports in the Enterprise Gateway configuration.



An Integration Server is not considered to be an Enterprise Gateway Server unless it has an enabled Enterprise Gateway external port. By default, this port will be disabled and all services, except for some basic services that Enterprise Gateway Server requires, will be set to deny.

The Enterprise Gateway external and registration ports work as a pair. One port is not functional without the other.

For instructions on setting up the Enterprise Gateway ports, see [“Configuring the Enterprise Gateway Ports”](#) on page 571.

Enterprise Gateway Rules and Alerts

You can configure Enterprise Gateway rules to filter requests that Enterprise Gateway Server sends to an Internal Server. You can also configure Enterprise Gateway Server to send an alert when a request violates a rule.

When a rule is configured to send an alert and a violation occurs, Enterprise Gateway Server logs the details in the server log and generates an alert. The alert message contains detailed information that includes the IP address from which the request was sent, user information, and the name of the rule filter that was met.

About Enterprise Gateway Rules

When a request violates a rule, Enterprise Gateway Server can deny the request, or it can allow the request and send an alert about the violation. This behavior is controlled by two types of Enterprise Gateway rules:

- **Denial rules.** Enterprise Gateway Server denies the request and sends an alert. The server stops processing the request as soon as it detects a match to a filter and does not consider other filters in the rule or any other rules for this request.
- **Alert rules.** Enterprise Gateway Server sends an alert about the violation and then continues processing the request. If a rule contains multiple filters, the server checks each filter and sends an alert each time it detects a match. After processing the rule, the server continues to the next rule. If there are no subsequent rules, or if the request does not violate any denial rules, the server allows the request.

Enterprise Gateway Server applies rules in the order in which they are displayed on the **Enterprise Gateway Rules** screen. Because a violation of a denial rule causes Enterprise Gateway Server to stop processing a request, it is important to prioritize the rules based on the order in which you want them to be evaluated. The server processes denial rules before alert rules.

You can configure an Enterprise Gateway rule to:

- Deny a request and send an alert about the violation.
- Allow a request and send an alert about the violation.
- Apply to all request types or only to SOAP, REST, or INVOKE requests.
- Apply to requests to use services or resources that have a particular name.
- Contain one more filters. If a request matches the condition specified by a filter, the request is in violation of the rule. For rules that contain multiple filters, if a request matches any one filter, the rule is in violation. For example, you can filter requests based on message size, the presence of an OAuth token, or the mobile application and type of device from which the request was sent.
- Contain no filters. Even if you do not specify filters in a rule, you can still use a rule to apply to certain request types or to requests for specific resources. For example, you can deny all SOAP requests.

For instructions on defining Enterprise Gateway rules, see [“Working with Enterprise Gateway Rules” on page 582](#).

About Enterprise Gateway Alerts

You can control the following aspects of alerts that Enterprise Gateway Server sends when a request violates a rule:

- Whether or not Enterprise Gateway Server issues an alert for a rule violation.
- How often Enterprise Gateway Server issues the alert. The server can issue an alert each time a rule is violated or at a specified time interval.

If you specify a time interval, Enterprise Gateway Server sends an alert when a request first violates the rule. If the violation occurs again within the specified time interval, the server waits until the end of the time interval before sending any more alerts for this request.

- The method Enterprise Gateway Server uses to send the alert. The server can send alerts by way of email or a flow service.

To send email alerts, the Integration Server acting as the Enterprise Gateway Server must be configured to send emails. Check the settings on the Settings > Resources screen under **Email Notification**. If your Enterprise Gateway Server is not configured to send emails, see [“Sending Messages About Critical Issues to E-mail Addresses” on page 220](#).

To send alerts by way of a flow service, Enterprise Gateway Server can use the `pub.security.enterpriseGateway:alertSpec` specification as the signature of the flow service. For more information about the specification, see *webMethods Integration Server Built-In Services Reference*.

- Whether a rule uses default alert options or its own customized alert options.

For instructions on configuring Enterprise Gateway alerts, see [“Specifying Alert Options” on page 589](#).

Version Interoperability Between Enterprise Gateway Server and Internal Server

Prior to version 9.7, Enterprise Gateway Server and Internal Server had to be the same version of Integration Server. However, beginning with version 9.7, this requirement is removed. Enterprise Gateway Server can now operate with previous versions of Integration Servers acting as Internal Servers. For example, Enterprise Gateway Server version 9.7 can work with Integration Server versions 9.7, 9.6, and 9.5 SP1 as the Internal Servers.

Interoperability between versions of Enterprise Gateway Server and Internal Server have the following benefits:

- You can upgrade only the Integration Server acting as Enterprise Gateway Server without having to upgrade the Integration Server acting as the Internal Server.
- You can leverage the features that are available in the latest releases of the Enterprise Gateway Server.

Keep the following information about Enterprise Gateway Server and Internal Server version interoperability in mind:

- The version of Enterprise Gateway Server must be equal to or higher than the version of Integration Server acting as the Internal Server.
- The version of Integration Server acting as the Internal Server must be 9.5 SP1 or later.
- You must obtain the correct webMethods Enterprise Gateway license from Software AG for the version of Enterprise Gateway that you want to run.
- If installing Enterprise Gateway for the first time, you must select the correct release version of webMethods Enterprise Gateway that you want to run from the list of available releases in the Software AG Installer regardless of the version of the Integration Server acting as the Internal Server. For example, to install Enterprise Gateway 9.7 you must select webMethods 9.7 as the **Release** version in Software AG Installer.
- You cannot upgrade Enterprise Gateway using the overinstallation procedure. To upgrade Enterprise Gateway, you must first install the Integration Server that you want to host Enterprise Gateway and then, install the higher version of Enterprise Gateway in the host Integration Server. For example, to upgrade Enterprise Gateway 9.6 that is hosted in Integration Server 9.6 to Enterprise Gateway 9.7, you must first install the host Integration Server 9.6. Then, you must run Software AG Installer again to install Enterprise Gateway 9.7 on the new installation of Integration Server 9.6.
- You must install Enterprise Gateway and Internal Server in different installation directories and preferably, on different machines. Do not configure a single Integration Server to be both an Enterprise Gateway Server and an Internal Server.
- Enterprise Gateway fixes are delivered as part of Integration Server fixes.
- If a version of Internal Server is not compatible with the Enterprise Gateway Server that you are configuring, Integration Server logs an error message stating the same in Enterprise Gateway Server as well as the Internal Server.
- Internal Server and Enterprise Gateway Server must use the same type of client authentication for the external port.

Integration Server 9.5 SP1 does not support using password digest for client authentication. If your Enterprise Gateway Server is 9.6 or higher, you cannot use digest authentication if your Internal Server is version 9.5 SP1.

Advantages of Enterprise Gateway over Traditional Third-Party Proxy Servers

An Enterprise Gateway configuration offers a number of advantages over traditional third-party proxy servers:

- Enterprise Gateway uses persistent connections. These connections eliminate the huge overhead of establishing SSL connections, while providing all the benefits of encryption.

- With Enterprise Gateway, you can configure your inner firewall to deny all inbound connections, isolating the Internal Server from the DMZ.
- Enterprise Gateway requires no changes to the external client.
- Enterprise Gateway rules can be defined to filter client requests.
- An Integration Server acting as an Enterprise Gateway Server can handle both HTTP and HTTPS requests. Typically, third-party proxy servers can handle only one or the other.

About Denial of Service Protection

You can use Enterprise Gateway to prevent Denial of Service (DoS) attacks. One form of DoS attack occurs when a client floods a server with many requests in an attempt to interfere with server processing. Using Enterprise Gateway, you can limit the number of requests that your Enterprise Gateway Server will accept within a specified time interval and the number of requests that it can process concurrently. By specifying these limits, you can protect your Enterprise Gateway Server and your Internal Server from DoS attacks.

You can configure Enterprise Gateway Server to consider the total number of requests from all IP addresses, or to consider the number of requests from individual IP addresses. For example, you might want to limit the total number of requests received to 10 requests in 10 seconds, and limit the number of requests coming from any single IP address to 2 requests in 10 seconds. When the server detects that a limit has been exceeded, the server sends an alert. Depending on your configuration, the server can temporarily block requests from all clients, or deny requests from particular IP addresses.

For instructions on configuring Enterprise Gateway Server to prevent DoS attacks, see [“Preventing Denial of Service Attacks”](#) on page 591.

About Trusted IP Addresses

To ensure that requests from trusted servers are not denied, you can configure a whitelist of IP addresses so that requests from these IP addresses are always allowed. While specifying trusted IP addresses, keep the following points in mind:

- Enterprise Gateway Server supports the inclusion of IPv4 and IPv6 addresses in the trusted IP addresses lists.
- You can specify a range of IP addresses using the classless inter-domain routing (CIDR) notation. To specify an IP address range, enter the first IP address in the range followed by a forward slash (/) and a CIDR suffix.

Example IPv4 address range:

- 192.168.100.0/22 represents the IPv4 addresses from 192.168.100.0 to 192.168.103.255
- 148.20.57.0/30 represents the IPv4 addresses from 148.20.57.0 to 148.20.57.3

Example IPv6 address range:

- `f000::/1` represents the IPv6 addresses from `f000::` to `ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff`.
- `2001:db8::/48` represents the IPv6 addresses from `2001:db8:0:0:0:0:0:0` to `2001:db8:0:ffff:ffff:ffff:ffff:ffff`.
- You can also specify a single IP address instead of an IP address range.
- You can specify multiple IP addresses or CIDR ranges. Separate each IP address or CIDR range with a comma (,).

For instructions on specifying trusted IP addresses, see [“Preventing Denial of Service Attacks” on page 591](#).

About Mobile Application Protection

You can use Enterprise Gateway to disable access for certain mobile application versions on a predefined set of mobile platforms. By disabling access to these versions, you are ensuring that all users are using the latest versions of the applications and taking advantage of the latest security and functional updates.

To use Enterprise Gateway application protection, you define mobile application filters in Enterprise Gateway rules. These filters check requests for application name, application version, and the type of device from which the request was sent. The mobile application provides this information in the request header. If a mobile application sends a request that matches a filter condition, Enterprise Gateway Server performs the configured action of denying the request or sending an alert about it. For instructions on configuring Enterprise Gateway Server to control mobile application use, see [“Controlling Use of Mobile Applications” on page 594](#).

About Mobile Data Synchronization

You can use the webMethods Mobile Support feature of Enterprise Gateway to synchronize data between mobile devices and backend applications. This feature consists of the following:

- An Integration Server package called `WmMobileSupport`, which contains the elements a business integration developer needs to create the server-side business logic of a data synchronization solution. This package is used with the Internal Server installed behind the firewall.
- A library called the Mobile Support Client, which contains APIs a mobile application developer needs to initiate data synchronization requests. This library resides on the system where mobile applications are developed.

For more information about using this feature to synchronize data between mobile devices and a backend application, see *Developing Data Synchronization Solutions with webMethods Mobile Support*.

About SQL Injection Protection

You can use the SQL injection protection filter of Enterprise Gateway to block requests that could possibly cause an SQL injection attack. When this filter is enabled in Enterprise Gateway, Integration

Server checks each request message for specific patterns of characters or keywords that are associated with potential SQL injection attacks. If a match is found in the request parameters or payload, Integration Server blocks the request from further processing.

Integration Server processes the incoming payload at Enterprise Gateway and if the incoming request contains characters that are not permitted for a database, based on how the rule is configured, Integration Server either denies the request and sends an alert about the violation or allows the request and sends an alert about the violation.

Integration Server provides two types of filters that you can enable to prevent SQL injection attacks:

- **Database-Specific SQL Injection Protection.** If enabled, Integration Server will check the incoming payload based on the specified database and GET or POST request parameters. If no parameter is specified, all input parameters will be checked for possible SQL injection attack. Integration Server adheres to the ESAPI (OWASP Enterprise Security API) standards while validating the parameters.

Parameters are applicable only for HTTP queries and HTTP form data, in which there are name value pairs.

For example, in the HTTP query string,

`http://localhost:1111/invoke/myjdbc.db:addUser?userid=' or '1'='1' --`, the parameter is `userid`.

- **Standard SQL Injection Protection.** If enabled, Integration Server will block XML and SOAP payload messages that contain quotation mark ('), number sign (#), or double hyphen (--) in the message.

For example, in the following XML payload, the Text elements (Title, Artist, and Country) contain invalid characters, ', #, and -- respectively.

```
<CATALOG>
<CD>
  <TITLE>Albu'm name</TITLE>
  <ARTIST>John# Smith</ARTIST>
  <COUNTRY>USA--</COUNTRY>
  <YEAR>2014</YEAR>
</CD>
</CATALOG>
```

About Antivirus Scan Filter

You can use the antivirus scan filter to configure Enterprise Gateway to interact with an *Internet Content Adaptation Protocol (ICAP)*-compliant server. An ICAP server is capable of hosting multiple services that you can use to implement features such as virus scanning or content filtering. Using the antivirus scan filter, Enterprise Gateway Server can leverage the ICAP protocol to scan all incoming HTTP requests and payloads for viruses.

Note:

The antivirus scan filter feature is certified on c-icap server, which is an implementation of an ICAP server, and can be integrated with all ICAP-compliant virus scanning applications.

Before enabling the antivirus scan filter, ensure that the following prerequisites are met:

- An ICAP-compliant server must be installed and configured in the DMZ and the Enterprise Gateway Server must be able to access the ICAP-compliant server.
- The ICAP-compliant server must have an ICAP service registered and the service must be accessible using the following format:

```
icap://<icap_server>:<icap_port>/serviceName
```

- Enterprise Gateway Server must be configured to send emails so that it can send alerts in case of any configuration or connectivity issues with the ICAP server. The email alerts are sent to the e-mail address of the administrator specified in the **Internal Email** field on the Settings > Resources screen.

If the antivirus scan filter is enabled as part of an Enterprise Gateway rule, Enterprise Gateway Server validates all incoming payloads by using the capabilities of the ICAP server in the following steps:

1. Enterprise Gateway Server requests that the ICAP server scan the request.
2. If the response from ICAP server includes a preview header, then Enterprise Gateway Server performs the following steps:
 - a. Enterprise Gateway Server responds with the amount data in bytes as specified in the preview header received from the ICAP server.
 - b. The ICAP server scans the preview content using the registered ICAP service.
 - c. If the ICAP server detects any malicious content in the request, depending on how the Enterprise Gateway rule is configured, Enterprise Gateway Server denies the request and sends an alert about the violation of the rule. Otherwise, Enterprise Gateway Server continues to send the rest of the file to the ICAP server to scan based on the server response.
3. If the response from ICAP server does not include a preview header, then Enterprise Gateway Server performs the following steps:
 - a. Enterprise Gateway Server requests that the ICAP server scan the entire request using the registered ICAP service.
 - b. If the ICAP server detects any malicious content in the request, depending on how the Enterprise Gateway rule is configured, Enterprise Gateway Server denies the request or allows the request and sends an alert about the violation of the rule.

Notes:

- Enterprise Gateway supports both REQMOD and RESPMOD methods based on the ICAP server response to the OPTIONS method.
- Enterprise Gateway Server sends a header named, X-wMUUID with every outbound ICAP request. The value of this header is a unique identifier and can be used to differentiate one scan from another.
- The Enterprise Gateway Server returns success if ICAP server status code is 200 and HTTP status code is within the range of 200-300. Enterprise Gateway Server returns failure if ICAP server status code is greater than or equal to 300.

About Custom Filter

You can use the custom filter to invoke a service that is available on the Enterprise Gateway Server.

You can use this capability to customize and invoke services in the Enterprise Gateway Server to perform actions such as custom authentication of external clients in the DMZ, logging or auditing in the DMZ, or implementation of custom rules for processing various payloads.

Using the custom service implementation, you can extract the HTTP headers and payload from a request and act on it as per your business requirements. Upon processing the headers, you can choose to forward the request to the internal server or deny the request and return an error message to the user.

Consider the following example: When a client request to the Enterprise Gateway Server contains only the host:port part of the request URL, the logon screen of the Internal Server appears by default. If you want to prevent access to the Internal Server through request URLs that contain only the host:port part, you can create an Enterprise Gateway rule with a custom filter to deny such requests.

You can also use the `pub.flow:setResponseHeaders` and `pub.flow:setResponseCode` services to add custom headers to the response and to set customized response codes.

Keep the following points in mind when creating an Enterprise Gateway rule with the custom filter:

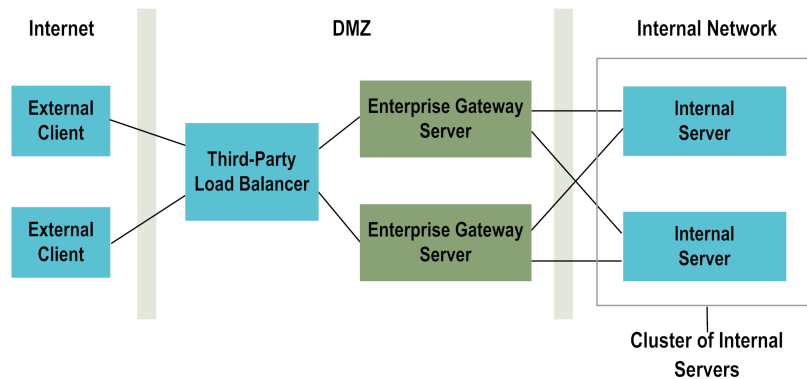
- Integration Server processes the incoming payload at Enterprise Gateway only if the custom logic in the service requires it to be processed.
- Integration Server duplicates the incoming payload of a request, by default and processes the complete request. When you create an Enterprise Gateway rule, clear the check box, **Clone the Payload** under **Custom Filter** so that Integration Server can process only the header information without duplicating the incoming payload.
- Use the `pub.security.enterprisegateway:customFilterSpec` specification as the signature of the custom service. For more details about the specification, see *webMethods Integration Server Built-In Services Reference*.
- The custom filter is the last filter that the Enterprise Gateway Server will check while processing an Enterprise Gateway rule.
- An Enterprise Gateway rule that contains the custom filter that is enabled must be a denial rule. Enterprise Gateway Server automatically converts an alert rule into a denial rule when you enable the custom filter.

Clustering in the Enterprise Gateway Configuration

You can have multiple Enterprise Gateway Servers and use a third-party product to load balance them. In addition, you can cluster your Internal Servers to improve availability, reliability, and scalability. For more information about Integration Server clustering, see *webMethods Integration Server Clustering Guide*.

The following diagram illustrates a supported configuration. Keep the following points in mind when working with clustered Internal Servers:

- Each Internal Server must connect to each Enterprise Gateway Server.
- Do not create a cluster of Enterprise Gateway Servers or include them as members of a cluster of “regular” Integration Servers.



Setting Up an Enterprise Gateway

This section describes the high-level steps for setting up the Enterprise Gateway. The following checklist summarizes these steps:

Done?	Task	Notes
	Install an Integration Server in your DMZ to be your Enterprise Gateway Server	<p>When you identify an Integration Server to be an Enterprise Gateway Server, keep in mind that any external client on the Internet can access this server. Therefore, be very security conscious about the services you make available and the users you define.</p> <p>Do not perform development work on this server and do not set up users or groups on it.</p> <p>Important: Do not configure a single Integration Server to be both an Enterprise Gateway Server and an Internal Server. This configuration is not supported, and unpredictable results will occur.</p>
	Disable the Developer and Replicator users	You will not need these users on an Enterprise Gateway Server. Disabling these users prevents someone from gaining access to your Enterprise Gateway Server through them. For more information, see “Disabling and Enabling User Accounts” on page 96.
	Configure the Enterprise Gateway external port	For instructions, see “Configuring the Enterprise Gateway Ports” on page 571.

Done?	Task	Notes
		<p>Note: If you plan to use an HTTPS port, you must store a server certificate, a server private key, and a CA certificate on this server. For instructions, see “Configuring Integration Server for Secure Communication” on page 457.</p>
	<p>Configure the Enterprise Gateway registration port</p>	<p>For instructions, see “Configuring the Enterprise Gateway Ports” on page 571.</p> <p>If you are going to set up an encrypted connection between the Internal Server and Enterprise Gateway Server, you can optionally store a certificate for the Internal Server’s administrator user on Enterprise Gateway Server. For more information, see “Importing a Certificate (Client or CA Signing Certificate) and Mapping It to a User” on page 519.</p> <p><i>Optional (but strongly recommended).</i> Set up IP address filtering on the registration port so that only the Internal Server can connect to Enterprise Gateway Server. This step provides an additional layer of protection to supplement the IP address filtering performed by your firewall and the user authentication.</p> <p>Note: Even if your external firewall filters out connections to the Enterprise Gateway registration port, IP address filtering is a good idea because it will stop insiders from connecting to Enterprise Gateway Server.</p> <p>For more information, see “Restricting IP Addresses that Can Connect to a Port” on page 487.</p>
	<p>Connect your Internal Server to Enterprise Gateway Server</p>	<p>For instructions, see “Connecting Your Internal Server to an Enterprise Gateway Server ” on page 576.</p>
	<p>Set values for the server configuration properties for Enterprise Gateway and Internal Server</p>	<p>Set values for or verify that the defaults for the following server configuration properties are suitable for your situation:</p> <ul style="list-style-type: none"> ■ watt.server.rg.internalregistration.timeout ■ watt.server.rg.internalsocket.timeout ■ watt.net.socketpool.sweeperInterval

Configuring the Enterprise Gateway Ports

The Enterprise Gateway external and registration ports work as a pair. One port is not functional without the other. Use this procedure to configure these ports on an Enterprise Gateway Server.

➤ To configure the Enterprise Gateway ports

1. Open the Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, on the **Security** menu, click **Ports**.
3. On the Security > Ports screen, click **Add Port**.
4. Under **Type of Port to Configure**, select **Enterprise Gateway Server**.
5. Click **Submit**.
6. On the **Edit Enterprise Gateway Server Configuration** screen, under **Enterprise Gateway External Port**, enter the following information:

For this parameter...	Specify...
Enable	Whether to enable or disable this port. If you choose to disable the port, you can enable it later on the Ports screen.
Protocol	The protocol to use for this port (HTTP or HTTPS). If you select HTTPS, additional security and credential boxes will be displayed at the bottom of the screen.
Port	The number you want to use for the external port. Use a number that is not already in use. This is the port that clients will connect to through your outer firewall.
Alias	An alias for the port. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.
Package Name	The package to associate with this port. You must specify the same package name for both external and registration ports. Typically, you will not need to work with packages on an Enterprise Gateway Server. Therefore, you can leave the default setting.

For this parameter... Specify...

Bind Address
(optional) The IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.

Backlog The number of requests that can remain in the queue for an enabled port before Enterprise Gateway Server begins rejecting requests. The default is 200. The maximum value is 65535.

Note:

This parameter does not apply to disabled ports. Enterprise Gateway Server refuses requests sent to disabled ports.

Keep Alive Timeout How long to wait before closing an idle connection to a client. The default is 20000 ms.

Threadpool Whether to create a private thread pool for this port or use the common thread pool.

- To have the server use the common server thread pool for this port, select **Disable**.
- To have the server create a private thread pool for this port so that it does not need to compete with other server functions for threads, select **Enable**.

If **Threadpool** is enabled, specify these additional parameters:

Threadpool Min

Minimum number of threads Enterprise Gateway Server maintains in this thread pool. When the server starts, the thread pool initially contains this minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed. The default is 1.

Threadpool Max

Maximum number of threads the server maintains in this thread pool. If this maximum number is reached, the server waits until services complete and return threads to the pool before running more services. The default is 5.

Threadpool Priority

Priority with which the JVM treats threads from this thread pool. The larger the number, the higher the priority. The default is 5.

Important:

For this parameter... Specify...

Use caution when setting the thread pool priority, as this setting can affect server performance and throughput.

When you view details for the port later, the server displays the total number of private threadpool threads currently in use for the port.

7. Under **Enterprise Gateway Registration Port**, enter the following information:

For this parameter... Specify...

Enable	Whether to enable or disable this port. If you choose to disable the port, you can enable it later on the Ports screen.
Protocol	The protocol to use for this port (HTTP or HTTPS). If you select HTTPS, additional security and credential boxes will be displayed at the bottom of the screen.
Port	The number you want to use for the registration port. Use a number that is not already in use. It is best <i>not</i> to use a standard port such as 80 (the standard port for HTTP) or 443 (the standard port for HTTPS) because the external firewall will allow access to those ports from the outside world.
Alias	An alias for the port. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.
Package Name	The package to associate with this port. You must specify the same package name for both external and registration ports. Typically, you will not need to work with packages on an Enterprise Gateway Server. Therefore, you can leave the default setting.
Bind Address (optional)	The IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.

8. For both external port and registration port, specify the type of client authentication to perform in the **Security Configuration** panel.

For external ports, this setting determines the type of authentication to perform for requests coming from the external client through the port. For registration ports, this setting determines the type of authentication to perform when the Internal Server establishes a persistent connection to Enterprise Gateway Server. Settings specified for registration ports control whether Enterprise Gateway Server will ask the Internal Server to present a certificate. See [“Authenticating Clients” on page 515](#) for more information about how clients are authenticated.

Note:

In a default Enterprise Gateway configuration, Enterprise Gateway Server does not perform client authentication. Rather, the server obtains authentication information (user/password or certificates) from the external client and passes this information to the Internal Server for authentication. However, you can have Enterprise Gateway Server perform client authentication as well. For details, see [“Performing Client Authentication on Enterprise Gateway Server” on page 581](#).

Select one of the following options:

Option	Description
Username/Password	Enterprise Gateway Server will not request client certificates. <ul style="list-style-type: none"> ■ For external ports, the server looks for user and password information in the header of requests coming from an external client. ■ For registration ports, the server looks for user and password information from the Internal Server.
Digest	For external ports, Enterprise Gateway Server uses password digest authentication. Enterprise Gateway Server looks for password digest information in the header of requests coming from an external client.
Request Client Certificates	Enterprise Gateway Server will request client certificates. <ul style="list-style-type: none"> ■ For external ports, the server requests client certificates for requests that come through this port. If the client does not present a certificate, the request proceeds using the user and password information contained in the request header. ■ For registration ports, the server requests a client certificate from the Internal Server. If the Internal Server does not present a certificate, the request proceeds using the user and password information.
Require Client Certificates	Enterprise Gateway Server will require client certificates. <ul style="list-style-type: none"> ■ For external ports, Enterprise Gateway Server requires client certificates for all requests that come through this port. If the client does not supply a certificate, the request fails.

Option	Description
	<p>Important: Use the same authentication mode here as you use for the Internal Server. For example, suppose you specify authentication mode Required on the Internal Server. Specifying Required on the Enterprise Gateway external port ensures that the request passed to the Internal Server includes a certificate.</p> <ul style="list-style-type: none"> ■ For registration ports, Enterprise Gateway Server requires a client certificate from the Internal Server. If the Internal Server does not supply a client certificate, the request fails. In addition, if the certificate is not mapped to a user with Administrator privileges on Enterprise Gateway Server, the request fails.
Request Kerberos Ticket	For external ports, Enterprise Gateway Server requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails.
Require Kerberos Ticket	For external ports, Enterprise Gateway Server looks for a Kerberos ticket from external clients. If the external client does not present a ticket, the request proceeds using the user and password information contained in the request header.
Use JSSE	If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure Socket Extension (JSSE) socket factory. If you set this value to No , the port supports only SSL 3.0 and TLS 1.0. The default is Yes .
	<p>Note: This field is available only if you selected HTTPS in the Protocol field.</p>

9. If you selected **HTTPS** in the **Protocol** field for either the external port or the registration port, optionally enter the following information under **Listener Specific Credentials**:

Note:
Use these settings only if you want to use a different set of credentials from the ones specified on the Certificates screen.

For this parameter...	Specify...
Keystore Alias	The keystore alias created for the keystore containing the certificate that Enterprise Gateway Server is to present to requests coming in through this port.
Key Alias	The alias for a specific key in the specified keystore.

For this parameter...	Specify...
Truststore Alias	The alias for the truststore file that contains the trusted root certificates associated with the CA signing authority.

10. Click **Save Changes**.

Deleting Enterprise Gateway External and Registration Ports

When you first set up the Enterprise Gateway external and registration ports, you associated each port with a package. If you delete one of the ports, Enterprise Gateway Server deletes the other port for you.

You delete ports on the Integration Server Administrator **Ports** screen. For details, see [“Configuring Ports” on page 149](#).

Connecting Your Internal Server to an Enterprise Gateway Server

This procedure describes how to connect your Internal Server to an Enterprise Gateway Server.

➤ To connect the Internal Server to the Enterprise Gateway Server

1. Open the Integration Server Administrator on the Integration Server acting as the Internal Server.
2. In the Navigation panel of the screen, on the **Security** menu, click **Ports**.
3. On the Security > Ports screen, click **Add Port**.
4. Under **Type of Port to Configure**, select **Internal Server**.
5. Click **Submit**.
6. On the **Edit Internal Server Configuration** screen, under **Internal Server**, enter the following information:

For this parameter...	Specify...
Enable	Whether to enable or disable this port. If you choose to disable the port, you can enable it later on the Ports screen.
Protocol	The protocol to use for this port (HTTP or HTTPS). If you select HTTPS, additional security and credential boxes will be displayed at the bottom of the screen.

For this parameter... Specify...

Package Name	The package to associate with the port. Typically, you will not need to work with packages on an Internal Server. Therefore, you can leave the default setting.
Alias	An alias for the port. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a-z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.
Max Connections	The number of connections maintained between Enterprise Gateway Server and the Internal Server. The default is 5.

Note:

For best performance, set the Max Connections setting on the listener to be slightly less than the Maximum Threads of the Server Threadpool setting on the Settings > Resources page. If you have more than one listener defined on the Internal Server, the sum of their Max Connection settings should be slightly less than the Maximum Threads of the Server Threadpool setting. Do not set Max Connections to be equal to the Internal Server's threadpool. Instead, reserve enough threads to handle the execution of scheduled services, triggers, and users that connect directly to the Internal Server.

Threadpool Whether to create a private thread pool for this port or use the common thread pool.

- To have the server use the common server thread pool for this port, select **Disable**.
- To have the server create a private thread pool for this port so that it does not need to compete with other server functions for threads, select **Enable**.

If **Threadpool** is enabled, specify these additional parameters:

Threadpool Min

Minimum number of threads the server maintains in this thread pool. When the server starts, the thread pool initially contains this minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed. The default is 1.

Threadpool Max

Maximum number of threads the server maintains in this thread pool. If this maximum number is reached, the server waits until

For this parameter... Specify...

services complete and return threads to the pool before running more services. The default is 5.

Threadpool Priority

Priority with which the JVM treats threads from this thread pool. The larger the number, the higher the priority. The default is 5.

Important:

Use caution when setting the thread pool priority, as this setting can affect server performance and throughput.

When you view details for the port later, the server displays the total number of private threadpool threads currently in use for the port.

7. Under **Enterprise Gateway Server**, enter the following information:

For this parameter...	Specify...
Host	The host name or IP address of the machine on which Enterprise Gateway Server is running.
Port	The port number of the registration port on Enterprise Gateway Server.

8. If you selected **HTTPS** in the **Protocol** box, optionally enter the following information under **Registration Credentials**. Note that the registration credentials specified here must match the settings on the Enterprise Gateway registration port:

For this parameter...	Specify...
User Name	The name of the user on Enterprise Gateway Server that the Internal Server should connect as.
Password	The password of the user on Enterprise Gateway Server that the Internal Server should connect as.
Use JSSE	If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure Socket Extension (JSSE) socket factory. If you set this value to No , the port supports only SSL 3.0 and TLS 1.0. The default is Yes .

Note:

This field is available only if you selected **HTTPS** in the **Protocol** field.

For this parameter...	Specify...
Keystore Alias	The keystore alias created for the keystore containing the certificate that the Internal Server sends to Enterprise Gateway Server for client authentication. The Internal Server sends this certificate when it makes its initial registration connection to Enterprise Gateway Server. The Internal Server sends this certificate only if asked to by Enterprise Gateway Server. Specify a value here only if you want to present a different server certificate from the one specified on the Certificates screen.
Key Alias	The key alias created for the key pair and associated certificate, in the previously specified keystore.
Truststore Alias	The alias for the truststore file that contains the trusted root certificates associated with the CA signing authority.

- Under **External Client Security**, in the **Client Authentication** list, specify the type of client authentication the Internal Server performs against external clients. External clients pass their authentication information to Enterprise Gateway Server, which in turn passes it to the Internal Server.

Option	Description
Username/Password	The Internal Server will not request client certificates from external clients. Instead it will look for user and password information in the request header.
Digest	The Internal Server will look for password digest information in the request header.
Request Client Certificates	The Internal Server will request client certificates for requests from external clients. If the external client does not present a certificate, the request proceeds using the user and password information contained in the request header.
Require Client Certificates	The Internal Server requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails.
Request Kerberos Ticket	The Internal Server requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails.
Require Kerberos Ticket	The Internal Server looks for a Kerberos ticket from external clients. If the external client does not present a ticket, the request proceeds using the user and password information contained in the request header.

Important:

Use the same authentication mode here as you use for the Enterprise Gateway external port. For example, specifying **Require Client Certificates** for both the Internal Server and the Enterprise Gateway external port ensures that the request passed to the Internal Server includes a certificate.

For more information about processing client certificates, see [“Authenticating Clients” on page 515](#).

10. Click **Save Changes**.

Viewing Connections to the Enterprise Gateway Registration Port

You can view a list of the Internal Server connections for a single Enterprise Gateway registration port.

➤ To view the connections to an Enterprise Gateway registration port

1. Open the Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel of the screen, on the **Security** menu, click **Ports**.
3. On the Security > Ports screen, click the port number of an Enterprise Gateway registration port.
4. On the Security > Ports > View Enterprise Gateway Server Details screen, click **Display Connections to Enterprise Gateway Registration Port**.

Integration Server Administrator displays the Security > Ports > Enterprise Gateway Registration Port Connections screen. The Enterprise Gateway Registration Port Connections: *portNumber* table lists the Internal Server connections to the Enterprise Gateway registration port at the specified *portNumber*. For each connection Integration Server Administrator displays the following information:

Field	Description
Host	IP address of the Internal Server connected to the Enterprise Gateway registration port.
Port	The local port number of the socket created for the remote Internal Server connection.
Connected (Number of Connections)	Indicates whether or not the Internal Server is connected to the Enterprise Gateway registration port.

Performing Client Authentication on Enterprise Gateway Server

In a default Enterprise Gateway configuration, external clients send requests to Enterprise Gateway Server, which in turn forwards authentication information (user/password or certificates) about these clients to the Internal Server. The Internal Server performs the authentication. This is the recommended configuration because certificates are safer when stored on the Internal Server, behind two firewalls.

However, if you want Enterprise Gateway Server to perform client authentication in addition to the authentication performed on the Internal Server, you can do so.

➤ To enable client authentication on Enterprise Gateway Server

1. In the Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server, navigate to the **Settings > Extended** screen and set the `watt.server.revInvoke.proxyMapUserCerts` system property to “true”.
2. If Enterprise Gateway Server is configured to request or require certificates, then for each external client to which you want to allow access, Enterprise Gateway Server must contain a copy of the client’s public certificate mapped to a user. For more information about mapping certificates, see [“Importing a Certificate \(Client or CA Signing Certificate\) and Mapping It to a User”](#) on page 519.

If, instead, Enterprise Gateway Server is configured to request certificates or perform authentication using user name and password, then Enterprise Gateway Server must contain a user name for that client.

Make sure that the external client’s imported certificate or user name is the same on both the Enterprise Gateway Server and the Internal Server.

3. Set the client authentication mode of the Enterprise Gateway external port to **Require Client Certificates**:
 - a. Navigate to the **Security > Ports** screen.
 - b. Find the row for the Enterprise Gateway external port. Click the port number, and then click **Edit HTTP Port Configuration**.
 - c. In the **Enterprise Gateway External Port** area of the Edit Enterprise Gateway Server Configuration screen, in the **Client Authentication** box, select **Require Client Certificates**.

Note:

Client authentication is supported only by the HTTPS protocol. If you do not see the **Client Authentication** box, change the external port protocol to **HTTPS**.

- d. Click **Save Changes**.

Working with Enterprise Gateway Rules

An Enterprise Gateway rule consists of one or more filters that determine which requests are allowed to pass through Enterprise Gateway Server to the Internal Server.

Creating an Enterprise Gateway Rule

Keep the following points in mind when creating an Enterprise Gateway rule:

- If you want to send email alerts, the Integration Server acting as the Enterprise Gateway Server must be configured to send email alerts. For more information, see [“Sending Messages About Critical Issues to E-mail Addresses”](#) on page 220.
- Be careful to prioritize rules in the order in which you want them to be evaluated. This is particularly important in the case of denial rules because a violation of a denial rule causes Enterprise Gateway Server to stop processing a request, sometimes before the entire rule has been evaluated.
- You can create a rule that contains no filters. You might use such a rule to deny all requests of a certain request type, or to send alerts about requests that use a particular resource.
- When a new rule is created, Enterprise Gateway Server applies the default alert options to the rule. To customize the alert options for a rule, see [“Specifying Rule-Specific Alert Options”](#) on page 591.
- Enterprise Gateway rules are categorized into denial and alert rules. When you create a new rule, it is added to the end of the list of rules in the appropriate category on the Security > Enterprise Gateway Rules page.

➤ To create an Enterprise Gateway rule

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. On the Enterprise Gateway Rules screen, click **Create Rule**.
4. In the **Rule Name** box, enter a name for the rule.

Valid rule names:

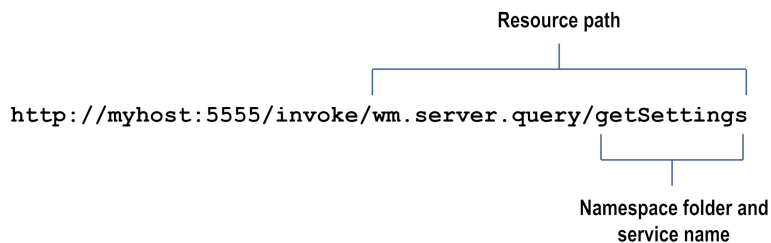
- Must be unique.
- Must not be null.
- Must not contain spaces.
- Must not contain these special characters:

? ~ ` ! @ # \$ % ^ & * () - + = { } | [] \ \ : \ " ; ' < > , /

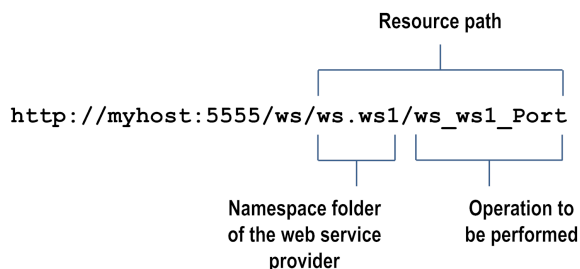
5. In the **Description** box, enter a brief description for the rule.
6. If you want to enable the rule immediately, select the **Enable** check box. Enterprise Gateway Server applies a rule to requests only if the rule is enabled. You can enable the rule later on the Enterprise Gateway Rules page.
7. From the **Request Type** list, select the type of request to which you want to apply the Enterprise Gateway rule. Select **ALL** if you want to apply this rule to all requests. Select **SOAP**, **REST**, or **INVOKE** to apply this rule to SOAP, REST, or INVOKE requests, respectively.
8. To filter a request based on the resource being requested, specify a string in the **Resource Path** box in the format *folder_name/service_name*. You can specify multiple paths, one to a line, and you can specify an asterisk (*) wildcard character. This box is available only if you selected a request type from the **Request Type** list.

Following are some examples of resource paths for INVOKE, web service, and REST requests.

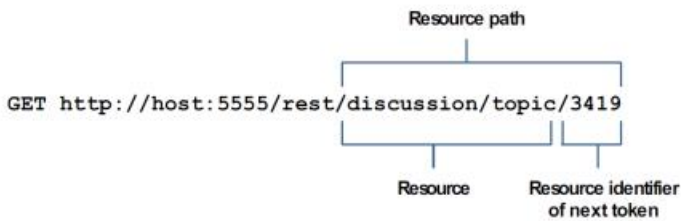
Example INVOKE request



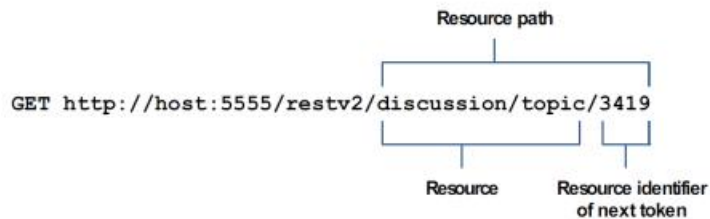
Example web service request



Example REST request - rest directive



Example REST request - restv2 directive



9. In the **Action** box, select one of the following:

Select...	To...
Deny Request and Alert	Deny the request if it violates the rule. Enterprise Gateway Server sends an alert based on the configured alert options.
Alert	Allow the request and send an alert based on the configured alert options.

10. In the **Error Message** box, enter a custom message that you want Enterprise Gateway Server to send to the client if a request violates an Enterprise Gateway deny rule. The server sends this message as a status message for an HTTP 403 status code, in the format "HTTP 403 *custom error message*."

11. Under **OAuth Filter**, select the **Require OAuth Credentials** check box if you want Enterprise Gateway Server to check for an OAuth token in the authorization header of the request. Select the **Enable** check box to enable this filter now, or enable this filter later by editing the rule on the Edit Rule screen.

12. Under **Message Size Filter**, in the **Maximum Message Size** box, enter the maximum size allowed for HTTP and HTTPS requests, in megabytes. If the request is larger than the size specified in this limit, the request will violate the rule and Enterprise Gateway Server will perform the configured action. Select the **Enable** check box to enable this filter now, or enable this filter later by editing the rule on the Edit Rule screen.

13. Under **Mobile Application Protection Filter**, specify one or more conditions that describe requests based on the device type from which the request was sent and the version of mobile

application that sent the request. If a request meets a condition, the rule is violated and Enterprise Gateway Server will perform the configured action. For example, you can specify that you will not allow requests from out-of-date versions of the ABC application sent from a ZZZ smart phone. Select the **Enable** check box to enable this filter now, or enable this filter later by editing the rule on the Edit Rule screen.

14. Under **SQL Injection Protection Filter**, select one or both of the following:

Select...	To...
Database-Specific SQL Injection Protection	<p>Block incoming requests that contain patterns recognized as potential SQL injection attacks for specific databases.</p> <ul style="list-style-type: none"> ■ From the Database list, select the database against which specific parameters needs to be checked. ■ Under Parameters, specify one or more GET or POST request parameters that could be present in the incoming requests. <p>Parameters can contain only alphanumeric characters, dollar sign (\$), and underscore (_).</p>
Standard SQL Injection Protection	<p>Block XML or SOAP payload messages that contain quotation mark ("), number sign (#), or double hyphen (--) anywhere within the message.</p>

Note:

Even though you can select both the filters, selecting multiple options may increase the message processing time because each filter requires a separate inspection of the request message.

Select the **Enable** check box to enable this filter now, or enable this filter later by editing the rule on the Edit Rule screen.

15. Under **Antivirus Scan Filter**, enter the following information if you want Enterprise Gateway Server to scan the incoming payload for viruses:

For this parameter...	Specify...
Antivirus ICAP Engine Name	A name for the ICAP server.
ICAP Host Name or IP Address	The host name or IP address of the machine on which the ICAP server is running.
ICAP Port Number	The port number on which the ICAP server is listening.

For this parameter...	Specify...
ICAP Service Name	The name of the service exposed by the ICAP server that you can use to scan your payload for viruses.

Select the **Enable** check box to enable this filter now, or enable this filter later by editing the rule on the Edit Rule screen.

- Under **Custom Filter**, click the **Browse** button adjacent to the **Invoke Service** field if you want to invoke a service that is available on the Enterprise Gateway Server. From the **Package Name** list, select the package that contains the service that you want to invoke. Enterprise Gateway Server populates the **Service Name** list with the names of the services that have the `pub.security.enterprisegateway:customFilterSpec` specification as its signature in the selected package. Select the service that you want to invoke.

In the **Run As User** box, click  to select the user name you want Enterprise Gateway Server to use to run the service. The default is Administrator.

You can select a user from the local or central directory. Enterprise Gateway Server runs the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.

Clear the **Clone the Payload** check box so that Integration Server does not duplicate the incoming payload and process only the header information in the request. When you select the check box, Integration Server duplicates the payload and processes the complete request. By default, the check box is selected.

Important:

If the custom filter is enabled, the Enterprise Gateway rule that contains the custom filter must be a denial rule. Enterprise Gateway Server automatically converts an alert rule into a denial rule when you enable the custom filter.

Select the **Enable** check box to enable this filter now, or enable this filter later by editing the rule on the Edit Rule screen.

- Click **Save Changes**.

Enterprise Gateway Server displays the rule on the Enterprise Gateway Rules page under **Denial Rules** or **Alert Rules** depending on the action you selected when you created the rule.

Enabling an Enterprise Gateway Rule

Enterprise Gateway Server applies a rule to requests only if the rule is enabled. You can enable a rule when you create it, or you can enable the rule later by using this procedure.

> To enable an Enterprise Gateway rule

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. On the Enterprise Gateway Rules screen, click **No** in the **Enabled** column for the rule you want to enable.
4. Click **OK** in response to the prompt asking to enable the rule.

When the rule is enabled, the server displays the ✓ icon and **Yes** in the **Enabled** column.

Disabling an Enterprise Gateway Rule

If you do not want to apply a specific Enterprise Gateway rule to requests, but you do not want to delete the rule, you can disable it.

Note:

Disabling a rule has no effect on its priority order on the Enterprise Gateway Rules screen.

➤ To disable an Enterprise Gateway rule

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. On the Enterprise Gateway Rules screen, click **Yes** in the **Enabled** column for the rule you want to disable.
4. Click **OK** in response to the prompt asking to disable the rule.

When the rule is disabled, the server displays **No** in the **Enabled** column.

Editing an Enterprise Gateway Rule

While editing an Enterprise Gateway rule, you can modify all properties of the rule except its name. If you change a denial rule to an alert rule or vice versa, Enterprise Gateway Server adds the rule to the bottom of the corresponding denial or alert list, thereby assigning the lowest priority to that rule in that category. To change the rule's priority, see [“Changing the Priority of an Enterprise Gateway Rule” on page 588](#).

➤ To edit an Enterprise Gateway rule


1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.

2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. On the Enterprise Gateway Rules screen, in the rules list, click the name of the rule that you want to edit.
4. In the properties screen for the selected rule, make the necessary modifications.
5. Click **Save Changes**.

Copying an Enterprise Gateway Rule

You can create a new Enterprise Gateway rule by copying an existing rule. When you copy a rule, the filter settings, alert options, and other properties, except for rule name, are copied.

> To copy an Enterprise Gateway rule

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. On the Enterprise Gateway Rules screen, click  in the **Copy** column for the rule you want to copy.

The server appends “_copy” to the rule name and displays the properties page of the rule.

4. View or modify the rule properties and filters as necessary. For more information, see [“Creating an Enterprise Gateway Rule” on page 582](#).
5. Click **Save Changes**.

Changing the Priority of an Enterprise Gateway Rule

Enterprise Gateway Server applies the Enterprise Gateway rules in the order in which they are displayed on the Enterprise Gateway Rules screen. If a request meets a filter condition in a denial rule, for example, the server denies the request and does not proceed to the next filter in the rule or to another rule. You can change the priority of the rules to ensure that they are applied in their intended order.

> To change the priority of an Enterprise Gateway rule

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.


- On the Enterprise Gateway Rules screen, in the **Priority** column for the rule you want to move, do one of the following:

To...	Click this button...
Move the rule up in the list	
Move the rule down in the list	

Deleting an Enterprise Gateway Rule

When you no longer need an Enterprise Gateway rule, you can delete it.

› To delete an Enterprise Gateway rule

- Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
- In the Navigation panel, select **Security > Enterprise Gateway Rules**.
- On the Enterprise Gateway Rules screen, click the  icon in the row that corresponds to the rule you want to delete.
- Click **Delete**. When you are prompted to confirm that you want to delete the rule, click **OK**.

Specifying Alert Options

You can specify default alert options that apply to all rules. You can also specify rule-specific alert options.

Specifying Default Alert Options

By default, if an Enterprise Gateway rule does not specify an alert type, Enterprise Gateway Server does not send an alert when a request violates the rule. If you want to issue an alert even when no alert type is specified for the rule, you can configure default alert options.

Note:

In order to send email alerts about rule violations, the Integration Server acting as the Enterprise Gateway Server must be configured to send emails. Check the settings on the Settings > Resources screen under **Email Notification**. If your Enterprise Gateway Server is not configured to send emails, see [“Sending Messages About Critical Issues to E-mail Addresses”](#) on page 220.

› To specify the default alert options


1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. In the Navigation panel on the Enterprise Gateway Rules screen, select **Default Alert Options** and then click **Edit Default Alert Options**.
4. From the **Alert Type** list, select one of the following options:

Select...	To...
None	Not send any alerts. This is the default.
Email	Send email alerts.
Flow Service	Invoke a flow service to alert you of a rule violation.

5. In the **Send Alert** box, do one of the following:
 - If you want Enterprise Gateway Server to send alerts every time a request violates a rule, select **On rule violation**.
 - If you want Enterprise Gateway Server to send alerts at specified intervals, select **Every** and then enter the time interval (in minutes). When a request violates the rule, the server sends an alert. If the violation occurs again within the specified time interval, the server waits until the end of the time interval before sending the alert. For example, suppose you specify the time interval as 1 minute. At 10:00, a request violates the rule and the server sends an alert. Ten seconds later, another request violates the rule. Ten seconds later, yet another request violates the rule. The server waits until 10:01 to send a summary of alerts that were generated in the time interval.
6. If you selected email as the alert type, enter the email addresses to which you want to send the alerts in the **Email Addresses** box. You can enter multiple email addresses. Separate addresses with semicolons (;).
7. If you selected flow service as the alert type, do the following:
 - a. In the **folder.subfolder:service** box, enter the fully qualified service name of the flow service you want Enterprise Gateway Server to execute.

Note:

Use the `pub.security.enterpriseGateway:alertSpec` specification as the signature of the flow service. For more details about the specification, see *webMethods Integration Server Built-In Services Reference*.

- b. In the **Run As User** box, enter the user name you want Enterprise Gateway Server to use when running the service. Click  to search for and select the user. You can select a user from the local or central directory.

Enterprise Gateway Server runs the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.

8. Click **Save Changes**.

Specifying Rule-Specific Alert Options

The default rule alert options are applied at a global level across all rules. You can also specify rule-specific alert options if you want a rule to have its own alert options.

➤ To specify rule-specific alert options

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. On the Enterprise Gateway Rules screen, in the list of rules, the **Alert Options** column shows whether the rule has the default alert options or customized alert options. Click **Default** or **Custom** to open the **Edit Alert Options** page.
4. Edit the alert options as required. For instructions on configuring alert options, see [“Specifying Default Alert Options” on page 589](#).
5. Click **Save Changes**.

Preventing Denial of Service Attacks

You can configure Enterprise Gateway Server to limit the number of requests the server will accept and the number the server will process concurrently. By doing so, you can protect both Enterprise Gateway Server and your Internal Server from Denial of Service attacks that use a flood of incoming requests to interfere with server processing.

You can configure whether Enterprise Gateway Server sets limits on requests globally, by IP address, or both.

However, to ensure that requests from trusted servers are not denied, you can specify a list of trusted IP addresses. Requests from these trusted IP addresses are always allowed. For more information about trusted IP address ranges, see [“About Trusted IP Addresses” on page 564](#).

Limiting Requests Globally

Use this procedure to limit the total number of requests Enterprise Gateway Server will accept and the number it will process concurrently. When you select the global option, the server considers all incoming requests, regardless of originating IP address. If you have configured a list of trusted IP addresses, requests from those IP addresses are not counted toward the request limit. When the number of requests exceeds the configured limit, the server blocks all requests at all Enterprise Gateway external ports for a period of time you specify. However, Enterprise Gateway Server will continue to accept requests from trusted IP addresses even after the configured limit is reached.

If you want to limit the number of requests Enterprise Gateway Server will accept from individual IP addresses, see [“Limiting Requests by IP Address” on page 593](#).

If you specify both the global and IP address options, Enterprise Gateway Server performs the global processing first.

> To limit requests globally

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. In the Navigation panel on the Enterprise Gateway Rules screen, click **Denial of Service Options** and then click **Configure Global Denial of Service**.
4. Select the **Enable** check box.
5. In the **Maximum Requests** box, enter the maximum number of requests you want Enterprise Gateway Server to accept in a given time interval. Then, enter the time interval, in seconds.
6. In the **Maximum Requests in Progress** box, enter the maximum number of requests that can be in progress at the same time.
7. In the **Block Interval** box, enter the number of minutes Enterprise Gateway Server is to block requests on every Enterprise Gateway external port.
8. In the **Error Message** box, enter a custom message to send to the client, if desired, when a request is denied.
9. In the **Trusted IP Address Range** box, enter the trusted IPv4 or IPv6 addresses so that requests from these IP addresses are always allowed. You can specify multiple IP addresses or IP address ranges separated by commas (.). For more information about trusted IP address ranges, see [“About Trusted IP Addresses” on page 564](#).
10. Click **Save Changes**.

Limiting Requests by IP Address

Use this procedure to limit the number of requests Enterprise Gateway Server will accept, or process concurrently, from any single IP address.

If the number of requests from an IP address exceeds the configured limit, Enterprise Gateway Server blocks requests from that IP address at all Enterprise Gateway external ports permanently, or for a period of time you specify. However, Enterprise Gateway Server will continue to accept requests from trusted IP addresses even after the configured limit is reached.

If you want to limit the total number of requests Enterprise Gateway Server will accept, regardless of IP address, see [“Limiting Requests Globally” on page 592](#).

If you specify both global and IP address options, Enterprise Gateway Server performs the global processing first.

➤ To limit requests by IP address

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.
3. In the Navigation panel on the Enterprise Gateway Rules screen, click **Denial of Service Options** and then click **Configure Denial of Service by IP Address**.
4. Select the **Enable** check box.
5. In the **Maximum Requests** box, enter the maximum number of requests that Enterprise Gateway Server can accept from a specific IP address in a given time interval. Then, enter the time interval, in seconds.
6. In the **Maximum Requests in Progress** box, enter the maximum number of requests that Enterprise Gateway Server can process concurrently from any single IP address.
7. In the **When Limit Exceeds** box, specify an action to take when the number of requests from a non-trusted IP address exceeds the specified limits.
 - To permanently deny future requests from the IP address, select **Add to Deny List**. Enterprise Gateway Server adds the IP address to the Enterprise Gateway deny list. As a result, requests from this IP address will be denied at every Enterprise Gateway external port. The IP address remains in the deny list until an administrator deletes it from the list.

Note:

The Enterprise Gateway deny list takes precedence over the port-level allow/deny list for the Enterprise Gateway external port.

- To temporarily block requests from this IP address, select **Block**. In the **Block Interval** box, specify the number of minutes you want requests to be blocked. To block requests, Enterprise Gateway Server adds the IP address to the Enterprise Gateway deny list. As a result, the server denies requests from this IP address on every Enterprise Gateway external port for the configured time period.
8. In the **Error Message** box, enter a custom message to send to the client, if desired, when a request is denied.
 9. In the **Trusted IP Address Range** box, enter IP addresses or range of trusted IPv4 or IPv6 addresses so that requests from these IP addresses are always allowed. You can specify multiple IP addresses or IP address ranges separated by commas (,). For more information about specifying trusted IP addresses, see [“About Trusted IP Addresses” on page 564](#).
 10. Click **Save Changes**.

Controlling Use of Mobile Applications

You can regulate requests from mobile applications by only allowing requests from certain versions of an application and from certain device types. By doing so, you ensure that all users are using the latest versions of mobile applications and taking advantage of the latest security and functional updates.

To control mobile application use, first you define a list of device types and a list of mobile applications you want to regulate. Then, you select from these values when you set up a mobile application filter in an Enterprise Gateway rule. The mobile application must provide the device type, application name, and application version in the request header in the following header fields:

Mobile-Device-Type

Mobile-Application-Name

Mobile-Application-Version

Note:

If a request includes a device type or application name that is not configured in an Enterprise Gateway rule, Enterprise Gateway Server will allow the request. Likewise, if a request specifies a version in an invalid format, the server will allow the request. A request violates a rule only when the request matches the condition specified in the filter.

➤ To control mobile applications

1. Open Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server.
2. In the Navigation panel, select **Security > Enterprise Gateway Rules**.

3. In the Navigation panel on the Enterprise Gateway Rules screen, select **Mobile Application Protection Options**.
4. Click **Edit Device Types** and enter the names of mobile devices, one per line, that you want to restrict. After you have entered the device types, click **Save Changes**.
5. Click **Edit Mobile Applications** and enter the names of mobile applications, one per line, that you want to restrict. After you have entered the device types, click **Save Changes**.
6. Click **Return to Enterprise Gateway Rules**.
7. On the Enterprise Gateway Rules screen, either select **Create Rule** to create a new rule or select an existing rule from the list.
8. On the Create Rule screen or the Edit Rule screen, scroll down to the **Mobile Application Protection Filter** portion of the screen.
9. Select a device type, a mobile application name, a condition, and an application version.

Specify the version in this format:

major-version.[minor-version.[sub-minor-version.[patch]]]

For example, to disallow requests from old versions (pre-3.0) of the BigApp mobile application sent from the XYZ mobile device, specify the following:

Field	Value
Device Type	XYZ
Mobile Application	BigApp
Condition	<
Mobile Application Version	3.0

To allow requests from all 3.0 versions (3, 3.0, 3.0.0, 3.0.0.0) of the BigApp mobile application sent from the XYZ mobile device, specify the following:

Field	Value
Device Type	XYZ
Mobile Application	BigApp
Condition	=
Mobile Application Version	3.0

10. To add another condition, click **Add** and repeat the previous step.
11. Click **Save Changes**.

Frequently Asked Questions about Enterprise Gateway

This section provides answers to some frequently asked questions about Enterprise Gateway.

If I define the Enterprise Gateway external port to use HTTPS, do I need to define the Enterprise Gateway registration port to be an HTTPS port too?

No. The external port and the registration port operate independently.

How many connections should I register between Enterprise Gateway Server and the Internal Server?

That depends on the expected load and the size of the transactions. A connection between Enterprise Gateway Server and the Internal Server is available except when a request is being written to the Internal Server or a response is being returned from the Internal Server. In other words, Enterprise Gateway connection utilization is I/O bound. Therefore, if you expect large, simultaneous transactions, increase the number of registered connections accordingly.

If the Internal Server runs out of registration connections, it will issue the following error message:

```
number requests waiting for a registration connection.
```

Each connection consumes a thread, either from the Internal Server's common thread pool or from the internal listener's private thread pool, if one is defined. The consumed thread can only be used to process requests from Enterprise Gateway Server.

If you have defined a private thread pool for the internal registration listener, the number of connections you can specify in the **Max Connections** box is limited to the maximum number of threads allowed in the private thread pool for this listener.

If you have multiple internal registration listeners, each with its own private thread pool, the same rule applies for each internal registration listener.

If you have not defined a private thread pool for an internal registration listener, a reasonable limit for the **Max Connections** box is 75% of the number of server threads specified in **Server Thread Pool Max Threads** box on the Settings > Resources page. If you have multiple internal registration listeners and none of them have private thread pools, the *sum* of all connections specified in the **Max Connections** boxes for these listeners should not exceed 75% of the number of server threads specified in **Server Thread Pool Max Threads**.

A thread will remain open unless it is closed by a firewall, a network glitch, or an exception.

Is there persistence with Enterprise Gateway Server?

No. Enterprise Gateway Server is just a network hop for the incoming request.

I want to authenticate the SSL credentials of external clients. Where do I set up certificates?

The following table shows where to set up certificates for the default Enterprise Gateway configuration, in which the Internal Server performs client authentication. If you want to perform client authentication on Enterprise Gateway Server as well, see [“Performing Client Authentication on Enterprise Gateway Server” on page 581](#).

Enterprise Gateway Server**Internal Server****Enterprise Gateway External Port**

- Set up a keystore that contains the server certificate and private key for Enterprise Gateway Server.
- Set up a truststore that contains the certificates of certificate authorities trusted by Enterprise Gateway Server.

Enterprise Gateway Server will make sure that certificates sent by external clients are signed by CAs in this truststore.

This truststore must be the same as the truststore on the Internal Server.

- (Optional) Import public certificates of external users and map them to users on Enterprise Gateway Server.

Do this only if you want to perform client authentication on Enterprise Gateway Server in addition to the Internal Server.

If you choose to perform client authentication on both Enterprise Gateway Server and the Internal Server, make sure the certificate mappings are the same on both servers.

- Set up a truststore that contains the certificates of certificate authorities trusted by the Internal Server.

The Internal Server will make sure that certificates sent by external clients (through Enterprise Gateway Server) are signed by CAs in this truststore.

This truststore must be the same as the truststore for the Enterprise Gateway external port.

- Import public certificates of external users and map them to users on the Internal Server.

If you choose to perform client authentication on both Enterprise Gateway Server and the Internal Server, make sure the certificate mappings are the same on both servers.

Enterprise Gateway Registration Port (HTTPS)

- Import the Internal Server's public certificate and map it to a user that has administrator privileges.
- Make sure the Internal Server's CA certificate is present in the truststore of the registration port
- Set up a keystore that contains the Internal Server's certificate and private key.
- Make sure the registration port's CA certificate is present in the Internal Server's truststore.

Can I use Enterprise Gateway Server as my outbound proxy server as well?

No. The only requests that go through Enterprise Gateway Server are inbound requests from the external client destined for the Internal Server and responses to those requests from the Internal Server back to the external client. Any unsolicited requests from the Internal Server go directly to the external client.

What authentication mode should I use for Enterprise Gateway Server and the Internal Server?

Authentication mode is the method a server uses to authenticate client requests. In a default Enterprise Gateway configuration, Enterprise Gateway Server receives authentication information from the external client and passes it to the Internal Server, which performs the authentication.

Be sure to specify the same authentication mode for the Internal Server and for the Enterprise Gateway external port. For example, if the Internal Server's authentication mode is **Required**, the external port must also be **Required** so that Enterprise Gateway Server always passes the external client's certificate to the Internal Server.

In contrast, the authentication mode of the Enterprise Gateway registration port does not need to match the authentication mode of the Internal Server or the Enterprise Gateway external port.

If you want to perform client authentication on Enterprise Gateway Server, see [“Performing Client Authentication on Enterprise Gateway Server” on page 581](#).

Does Enterprise Gateway support the FTP protocol?

No, support is limited to HTTP and HTTPS only.

Are the SOCK and SSLSOCK protocols supported?

No, these were proprietary protocols used in older releases. Starting with the 7.1 release, SOCK and SSLSOCK have been replaced by HTTP and HTTPS.

Is it possible to filter requests on Enterprise Gateway Server?

Yes. You can use Enterprise Gateway rules to filter requests based on a number of factors, including request size, request type, and the name of the resource being invoked.

After an interruption in the network connection between Enterprise Gateway Server and the Internal Server, the ports on the Internal Server must be re-enabled. How can this be prevented?

On the Internal Server, change the value of the `watt.server.rg.internalregistration.timeout` to be something other than the default value of 0, which means that the Internal Server never closes an unresponsive connection to the Enterprise Gateway Server. Additionally, make sure the value of `watt.server.rg.internalregistration.timeout` on the Internal Server is greater than the value of `watt.net.socketpool.sweeperInterval` on the Enterprise Gateway Server.

How can a client request on the Enterprise Gateway Server be prevented from waiting indefinitely for a connection to the Internal Server?

Enterprise Gateway Server can pass a client request to an Internal Server only when a connection to the Internal Server is available. If there are no connections to the Internal Server or if all of the Internal Server connections are busy handling other requests, a client request received through the external port of the Enterprise Gateway Server is placed in a waiting state. It is possible that several minutes could elapse before a connection becomes available which makes it possible that the client times out the request. Because the client does not receive any feedback from the Enterprise Gateway Server during this waiting period, a retry mechanism on the client side could resend the client request, resulting in duplicate transactions.

Integration Server includes a server configuration property `watt.server.rg.internalsocket.timeout` that controls how long the Enterprise Gateway Server waits for a connection to the Internal Server and returns a HTTP 500-Internal Server Error to the requesting client. If a connection to the Internal Server becomes available within the specified timeout period, Enterprise Gateway Server forwards the request to the Internal Server. If a connection does not become available

before the timeout elapses, Enterprise Gateway Server returns a HTTP 500-Internal Server Error to the requesting client.

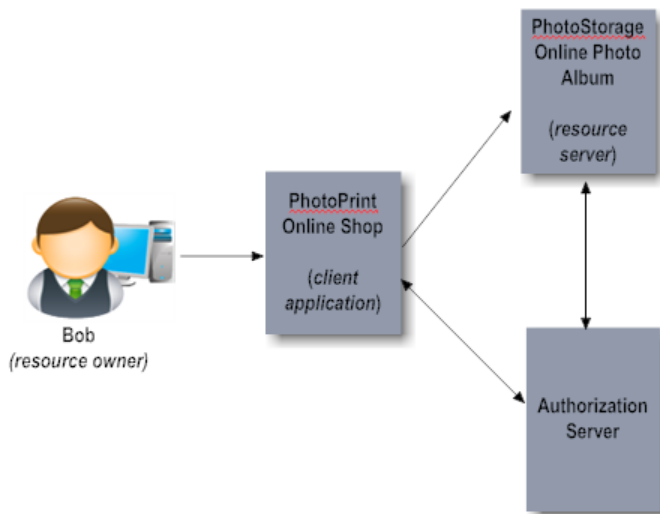
33 Configuring OAuth

■ What Is OAuth?	602
■ Using OAuth with Integration Server	602
■ OAuth Client Types	604
■ Authorization Grant Types Supported by Integration Server	604
■ The Integration Server OAuth Services	610
■ Important Considerations for Using OAuth Features	612
■ Configuring Integration Server for OAuth	612
■ About Using Integration Server as the Resource Server	630
■ Using an External Authorization Server	630

What Is OAuth?

The OAuth 2.0 Authorization Framework (OAuth) facilitates the sharing of private resources (data or services) with a third-party *client application* (*client*). In an OAuth session, private resources are stored on a *resource server* and the owner of the resources, or *resource owner*, grants the client application permission to access them. The resource owner is typically a person; however, in some cases it could be an application. When a resource owner grants permission, the OAuth *authorization server* issues an *access token* to the client application. When the client application passes the access token to the resource server, the resource server communicates with the authorization server to validate the token and, if valid, provides access to the resources.

The following example illustrates the roles involved with an OAuth session. In the example, Bob is the resource owner who wants to access and print his photos stored on the PhotoStorage website (the resource server) using the PhotoPrint service (the client application). PhotoPrint supplies Bob with an application that runs on his device (phone, laptop, etc). Bob uses that application to initiate the process. PhotoPrint sends a request to the PhotoStorage authorization server. The authorization server requests authorization from Bob and issues a token to PhotoPrint. PhotoPrint can then access Bob's photos on PhotoStorage.



An in-depth description of OAuth is beyond the scope of this guide but is available elsewhere. For information about the OAuth protocol, see the OAuth 2.0 Authorization Framework.

Using OAuth with Integration Server

Integration Server can be an OAuth client, an authorization server, or a resource server. Integration Server Administrator provides an OAuth configuration feature that developers can use to create and manage these roles. To enable the OAuth features, your Integration Server must be equipped with a license from Software AG. For information about licensing, see [“Important Considerations for Using OAuth Features”](#) on page 612.

Integration Server as an OAuth Client

When Integration Server is an OAuth client application, you use the `pub.client.oauth.executeRequest` service to access resources from providers such as Facebook, Google, Twitter, or Integration Server. For information about accessing resources using the `pub.client.oauth.executeRequest` service, see *webMethods Integration Server Built-In Services Reference*.

Integration Server as an Authorization Server

When Integration Server acts as an authorization server, it receives authorization requests from client applications. Client applications initiate the request by invoking the `pub.oauth.authorize` service. The authorization server handles the interactions between the client application, resource server, and resource owner for approval of the request. For information about configuring Integration Server as an authorization server, see [“Configuring Integration Server for OAuth” on page 612](#).

When Integration Server acts as an authorization server, it issues access tokens as *bearer tokens*. A bearer token is an access token that allows any party in possession of the access token (Bearer) to use the token. The authorization server retains the information about the bearer tokens it issues, including the user information. When the client presents a bearer token to the resource server, the resource server sends the token to the authorization server to ensure that the token is valid and that the requested service is within the scope for which the access token was issued. A *scope* is the definition of the folders and services (resources) that the client can access on behalf of a resource owner.

If the user is authorized to access the folders and services, the resource server executes the request. If the user does not have privileges to access the resources, the resource server rejects the request. For information about user privileges, see [“Managing Users and Groups” on page 83](#).

Integration Server as an External Authorization Server

Access tokens generated by Integration Server’s authorization server can be used to access resources in other vendors’ servers. That is, an Integration Server authorization server can respond to a requests for access token introspection from a resource server that is not an Integration Server. This is part of Integration Server support for RFC 7662, OAuth 2.0 Token Introspection.

The `pub.oauth*` services provide clients with services for interacting with an Integration Server used as an OAuth 2.0 authorization server, including services for requesting a token, determining whether a token is still active, refreshing an access token, and revoking an access token. The introspection endpoint for the Integration Server authorization server is the following URL:

```
https://host:port/invoke/pub.oauth/introspectToken
```

This invokes the `pub.oauth:introspectToken` service on the Integration Server authorization server located at `host:ports`.

For more information about built-in services for OAuth see, *webMethods Integration Server Built-In Services Reference* .

Integration Server as a Resource Server

When Integration Server acts as a resource server, it receives requests from client applications that include an access token. The resource server asks the authorization server to validate the access token and user. If the token is valid and the user has privileges to access the folders and services, the resource server executes the request. The resource server and the authorization server might be the same Integration Server instance or might be different Integration Server instances. The authorization server might also be a third party authorization server. For information about using Integration Server as a resource server, see [“About Using Integration Server as the Resource Server” on page 630](#). For information about configuring an third party authorization server, see [“Using an External Authorization Server” on page 630](#).

OAuth Client Types

OAuth 2.0 Authorization Framework defines two types of clients.

- A *confidential client* is a client that supplies a client ID and client secret to the authorization server in order to obtain an access token. Confidential clients correspond to an account on the authorization server. If a client does not have the proper credentials (client ID and secret) for the user account, the authorization server does not grant the client an access token.

To specify a confidential client in Integration Server Administrator, set **Type** to **Confidential**. For more information, see [“Registering Clients” on page 616](#).

- A *public client* is a client that uses only a client ID for identification, with no other credentials. Public clients are typically implemented in a browser using a scripting language such as JavaScript. Because the authorization server does not require any other credentials, the authorization server grants *any* client with a valid client ID an access token.

To specify a public (implicit) client in Integration Server Administrator, set **Type** to **Public**. For more information, see [“Registering Clients” on page 616](#).

Authorization Grant Types Supported by Integration Server

The flow of authorization requests and responses between the resource owner, client application, authorization server, and resource server depends on the authorization grant type defined by the OAuth session. Integration Server supports the following authorization grant types:

- Authorization code grant
- Implicit grant
- Resource owner password credentials grant
- Client credentials grant

For each OAuth client registered with an Integration Server, you must specify the OAuth grant types that a client can use.

Authorization Code Grant

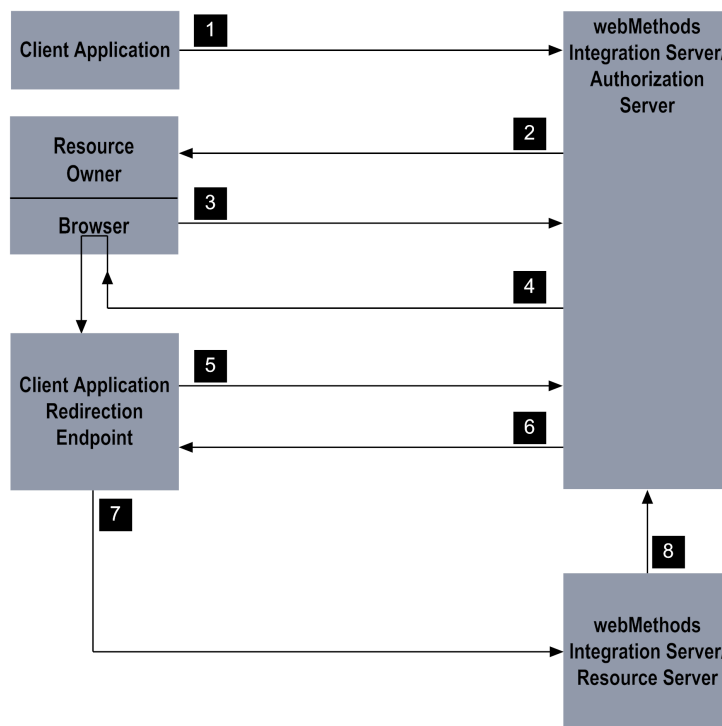
The authorization code grant type is used to authenticate and provide access to clients that have credentials on the authorization server. This grant type requires the client to authenticate to the authorization server before obtaining an access token.

You use the authorization code grant type to authenticate and provide access to confidential and public clients.

When using the authorization code grant type, the authorization server can issue a *refresh token* to the client application along with the access token. A refresh token enables clients to get a new access token without requesting additional approval from the resource owner. When the access token expires, the client application can use the `pub.oauth:getToken` service to pass the refresh token to the authorization server to request a new access token.

The following diagram illustrates how the Integration Server authorization server participates in the authorization code grant process.

Authorization Code Grant Flow



The following table describes each step in the authorization code grant flow.

Stage	Description
1	The client application initiates the process by calling the <code>pub.oauth:authorize</code> service to request access to the resource owner's data.

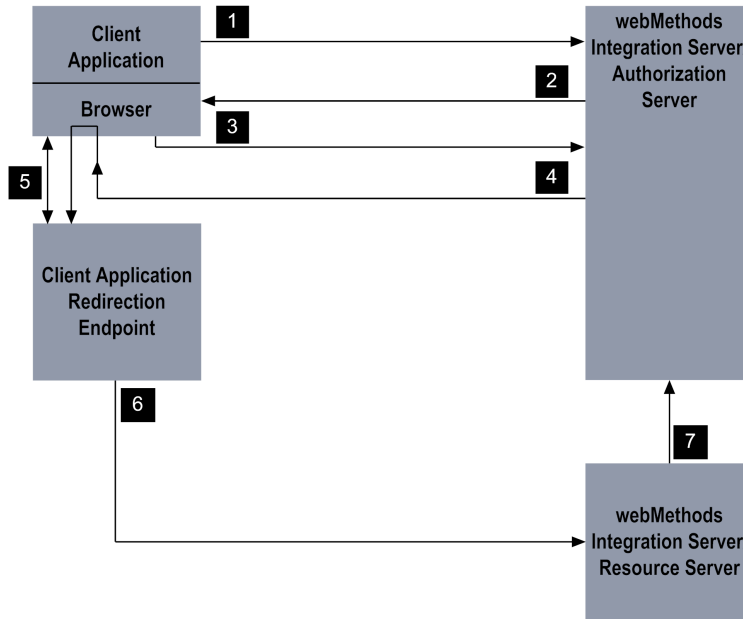
Stage	Description
2	The <code>pub.oauth:authorize</code> service validates the request. If valid, the service responds with an HTML page that informs the resource owner that the client application is requesting access within a specified scope. The resource owner uses the HTML page to approve or deny the request.
3	When the resource owner approves the request, the approval page invokes an internal service on Integration Server. If the resource owner denies the request, an error is returned.
4	Integration Server generates an authorization code for the client application. The server uses HTTP redirection to instruct the resource owner's browser to transmit the authorization code to the redirection URI provided by the client application.
5	The service at the client application's redirection URI passes the authorization code to the <code>pub.oauth:getToken</code> service on Integration Server to exchange the authorization code for an access token. Integration Server also ensures that the client is allowed to use the authorization code grant.
6	Integration Server issues an access token to the client application. If set to do so, the authorization server also issues a refresh token to the client.
7	The client application uses the access token to request a resource on the resource server.
8	The resource server checks with authorization server to make sure the access token is valid. If it is, the resource server checks that the requested service is within the scope for which the access token was issued. It also checks that the resource owner is authorized to access the requested resource.

Implicit Grant

The implicit grant type is used to authenticate browser-based applications and mobile applications. This grant type is less secure than the authorization code grant because it does not require the client to authenticate with the authorization server. In addition, the implicit grant type passes the access token through the resource owner's browser, exposing it to theft by malicious applications on the resource owner's device. The following diagram illustrates how the Integration Server authorization server participates in the implicit grant process.

You use the implicit grant type to authenticate and provide access to public clients.

Implicit Grant Flow



The following table describes each step in the implicit grant flow.

Stage	Description
1	The client application initiates the process by calling the <code>pub.oauth:authorize</code> service to request access to the resource owner's data.
2	The <code>pub.oauth:authorize</code> service validates the request. If valid, the service responds with an HTML page that informs the owner that the client application is requesting access within a specified scope. The resource owner uses a form on the HTML page to approve or deny the request.
3	When the resource owner approves the request, the approval page invokes an internal service on Integration Server. If the resource owner denies the request, an error is returned.
4	Integration Server redirects the resource owner's browser to the client application's redirection endpoint, appending the access token as a fragment on the redirection URI.
5	The resource owner's browser holds the access token in memory until the client application retrieves the access token.
6	The client application uses the access token to request a resource on the resource server.
7	The resource server checks with authorization server to make sure the access token is valid. If it is, the resource server makes sure that the requested service is within the scope for which the access token was issued. It also checks that the resource owner is authorized to access the requested resource.

Resource Owner Password Credentials Grant

The resource owner password credentials (ROPC) grant allows resource owners to provide trusted clients with their credentials which the clients can then use to obtain an access token. Clients can use the access token to access resources on the resource server. The intent of the ROPC grant is to provide access to resources in situations where there is an existing trust relationship between the resource owner and the client or when it is not possible to use other authorization grant types.

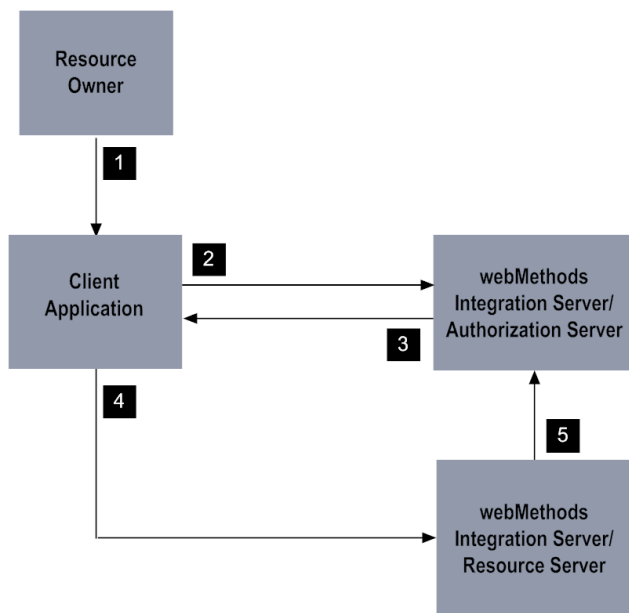
With the ROPC grant type, the client does not need resource owner approval nor does the client need a redirection endpoint. Instead the client calls the token endpoint on the authorization server directly, presenting the resource owner's credentials for authentication. The token endpoint authenticates the credentials and issues an access token.

When using the ROPC grant type, the authorization server can issue a *refresh token* to the client application along with the access token. A refresh token enables clients to get a new access token without requesting additional approval from the resource owner. When the access token expires, the client application can use the `pub.oauth:getToken` service to pass the refresh token to the authorization server to request a new access token.

You use the ROPC grant type to authenticate and provide access to confidential clients and public clients.

The ROPC grant type is the least secure of all the grant type supported by Integration Server.

Resource Owner Password Credentials Grant Flow



The following table describes each step in the ROPC grant flow.

Stage	Description
1	The resource owner provides their credentials to the client application

Stage	Description
2	The client application presents the resource owner's credentials to the token endpoint, the <code>pub.oauth:getToken</code> service, on the authorization server.
3	On the authorization server, the <code>pub.oauth:getToken</code> service authenticates the provided credentials and verifies that the credentials belong to a confidential client allowed to use the resource owner password credentials (ROPC) grant. The service also checks that the scopes specified with the client request are allowed for the client. The authorization server returns an access token to the client. If set to do so, the authorization server also returns a refresh token to the client.
4	The client application uses the access token to request a resource on the resource server.
5	The resource server checks with authorization server to make sure the access token is valid. If it is, the resource server makes sure that the requested service is within the scope for which the access token was issued. It also checks that the resource owner is authorized to access the requested resource.

Client Credentials Grant

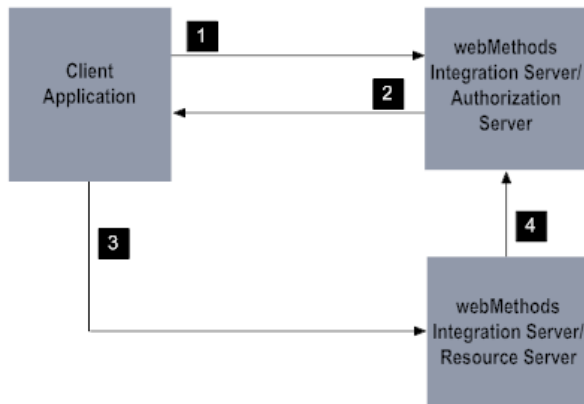
The client credentials grant is used to authenticate and provide access to clients that have credentials on the authorization server. The client credentials grant allows the client to use their credentials to obtain an access token that can be used to access resources. The client credentials grant is often used to access resources that are already under the control of the client, or when the resource owner trusts the client to access its resources, or for machine-to-machine interactions. An example of machine-to-machine interaction might be a background process that runs on the resource owner's device and the background process needs access to resources protected by OAuth.

Unlike the authorization code grant type, a client using the client credentials grant does not need resource owner approval before obtaining an access token, nor does the client need a redirection endpoint. Instead, the client presents credentials directly to the token endpoint and requests an access token. The token endpoint authenticates the credentials and issues an access token. The token endpoint does not issue a refresh token as refresh tokens are not supported by the client credentials grant.

The client credentials grant type is less secure than the authorization code grant type.

A client that uses the client credentials grant must have credentials on the authorization server which means the client must be a confidential client. When you create a client that will use the client credentials grant, you must give the client permission to access the resources it needs. This is different than the other grant types where Integration Server knows the resource owner because the resource owner either provided its credentials when approving the request (authorization code and implicit grants) or the client supplies the resource owner's credentials (ROPC grant). With the client credentials grant, Integration Server does not know the owner because the client does not need resource owner approval. Consequently, it is the client who must have permission to access the resources. You must add the client to one or more user groups that have the access permissions that the client needs.

Client Credentials Grant Flow



The following table describes each step in the client credentials grant flow.

Stage	Description
1	The client application initiates the process by presenting its credentials to the token endpoint, the <code>pub.oauth:getToken</code> service, on the authorization server.
2	On the authorization server, the <code>pub.oauth:getToken</code> service authenticates the client and verifies that the credentials belong to a confidential client allowed to use the client credentials grant. The service also checks that the scopes specified with the client request are allowed for the client. The authorization server returns an access token to the client.
<p>Note: A refresh token is not returned for a client credentials grant.</p>	
3	The client application uses the access token to request a resource on the resource server.
4	The resource server checks with authorization server to make sure the access token is valid. If it is, the resource server checks that the requested resource is within the scope for which the access token was issued.

The Integration Server OAuth Services

Integration Server provides public services for use with OAuth in the `WmPublic` package.

The following table lists services that you use to authorize a client application to access a resource server.

Service	Description
<code>pub.oauth:authorize</code>	Initiates an authorization request from the client application to the Integration Server authorization server.

Service	Description
pub.oauth:getAccessToken	<p>Requests an access token from the Integration Server authorization server. The request includes the authorization code sent to the redirection URI by the authorization server.</p> <p>The authorization server validates the request and generates an access token and refresh token (optional). The tokens, along with the client identifier, expiration time, and scope are stored in the authorization server's cache.</p> <p>Note: The pub.oauth:getAccessToken service is deprecated. Use pub.oauth:getToken instead.</p>
pub.oauth:getToken	<p>Requests an access token from the Integration Server authorization server. The request includes the authorization code sent to the redirection URI by the authorization server.</p> <p>The authorization server validates the request and generates an access token and refresh token (optional). The tokens, along with the client identifier, expiration time, and scope are stored in the authorization server's cache.</p> <p>Note: The pub.oauth:getToken service replaces the pub.oauth:getAccessToken and pub.oauth:refreshAccessToken services which are deprecated.</p>
pub.oauth:introspectToken	<p>Checks whether an access token or refresh token generated by an Integration Server used as an authorization server is active.</p>
pub.oauth:refreshAccessToken	<p>Sends a request to the Integration Server authorization server to refresh the access token.</p> <p>Note: The pub.oauth:refreshAccessToken service is deprecated. Use pub.oauth:getToken instead.</p>
pub.oauth:removeExpiredAccessTokens	<p>Removes expired access tokens from the database.</p>
pub.oauth:revokeToken	<p>Revokes a token on an Integration Server authorization server.</p>

For more information about using the Integration Server OAuth services, see *webMethods Integration Server Built-In Services Reference* .

Important Considerations for Using OAuth Features

Keep the following points in mind when using OAuth features:

- If you are using webMethods Enterprise Gateway to process requests from external clients, keep in mind that OAuth configuration settings in Integration Server Administrator (**Security > OAuth**) are only available on Integration Servers that do not act as an Enterprise Gateway Server. The Internal Server that sits behind the firewall processes all OAuth requests. For more information about webMethods Enterprise Gateway, see [“Configuring webMethods Enterprise Gateway” on page 557](#).
- In order for Integration Server to log OAuth activity, the Security logger must be enabled and configured to log the following security areas: Authentication and Authorization. For detailed instructions on enabling the security logger and selecting security areas to audit, see the *webMethods Audit Logging Guide*.

Configuring Integration Server for OAuth

Before you can begin to use OAuth in your Integration Server environment, you need to specify settings for the authorization server and/or resource server, depending on which role the Integration Server plays in your OAuth solution. When the authorization server and resource server are configured, you can start registering clients and managing your OAuth scopes.

The following table describes the basic stages for configuring Integration Server for OAuth.

Stage 1 Configure OAuth settings.

During this stage, you configure the OAuth settings on Integration Server. Integration Server is configured to use certain OAuth settings by default. For information about configuring these settings to reflect those for your system, see [“Configuring OAuth Settings” on page 613](#).

Note:

This stage primarily applies to an Integration Server being used as an authorization server. However, if the Integration Server is acting as the resource server, you must use the **Authorization server** field on the Security> OAuth > Edit OAuth Global Settings page to identify the authorization server for the resource server.

Stage 2 Define clients.

During this stage, you define the clients that are authorized to access the authorization server and specify which grant types they can use. For information about registering, modifying, and deleting clients, see [“Defining Clients” on page 616](#).

When using Integration Server as the authorization server, the Integration Server and the resource server need to have the same resource owners. This

requirement does not apply when using an external authorization server or if all of your clients will use the client credentials grant type.

If you are using Integration Servers for the authorization and resource servers, you can define the `client_id` values on one Integration Server and then deploy the values to the other Integration Server.

For clients that use the client credentials grant, the user accounts associated with the clients need to be on the authorization server and the resource server.

Stage 3 Define scopes.

During this stage, you define the scopes available for the clients to access. For information about adding, modifying, and deleting scopes, see [“Defining Scopes” on page 622](#).

Your authorization server and resource server must have the same scope names. You can define the scope names on each server. Or, if you are using Integration Servers for the authorization and resource servers, you can define the scopes on one Integration Server and then deploy the values to the other Integration Server.

Stage 4 Associate scopes to clients and vice versa.

During this stage, you associate scopes to clients. When you associate scopes and clients, you authorize the scopes that each client can access. For information about adding, removing, and viewing the associations between scopes and clients, see [“Associating Scopes and Clients” on page 624](#).

Note:

This stage applies to an Integration Server being used as an authorization server only. You do not need to complete this stage for an Integration Server being used as a resource server.

Stage 5 If you want to prevent specific client applications from accessing resources *after* the authorization server has granted an access token, you can do either of the following:

- On the authorization server, delete the active access and refresh tokens granted for that client application. For information about viewing and deleting tokens, see [“Managing Tokens” on page 628](#).
- On the resource server, disable the client application. For information about disabling client applications, see [“Enabling and Disabling Clients” on page 620](#).

Configuring OAuth Settings

The OAuth global settings for the authorization server control whether HTTPS is required for OAuth communications. You can also specify global values for authorization code and access

token expiration intervals. The expiration intervals can be set globally or configured for each individual client.

➤ **To configure the OAuth settings**

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Edit OAuth Global Settings**.
4. Under Authorization Server Settings, complete the following fields for when Integration Server acts as the authorization sever.

Field	Description				
Require HTTPS	<p>Indicates whether the authorization server should require an HTTPS connection to authorize requests.</p> <p>If enabled (the default), Integration Server requires that the authorization server uses HTTPS to invoke the pub.oauth services. If disabled, Integration Server allows client applications to use HTTP to access the pub.oauth services.</p> <p>If Require HTTPS is enabled and the client application accesses any of the pub.oauth services over HTTP, Integration Server issues an HTTP 400 error response to the client and writes a service exception to the error log.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: You can disable Require HTTPS to simplify development, but you should use HTTPS in production in accordance with the OAuth Framework. If you do not require HTTPS, the authorization server transmits access tokens in clear text, making them vulnerable to theft.</p> </div>				
Authorization code expiration interval	<p>Specifies the length of time (in seconds) that the authorization code issued by the authorization server is valid.</p> <p>Valid values are between 1 and 2147483647. The default value is 600.</p>				
Access token expiration interval	<p>Specifies the length of time (in seconds) that access tokens issued by the authorization server are valid.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Select</th> <th style="text-align: left;">To.</th> </tr> </thead> <tbody> <tr> <td>Never Expires</td> <td>Indicate that the access token never expires</td> </tr> </tbody> </table>	Select	To.	Never Expires	Indicate that the access token never expires
Select	To.				
Never Expires	Indicate that the access token never expires				

Field	Description						
	<p>Expires in and enter the Specify the length of time that the access number of seconds. The token is valid maximum value is 2147483647. The default is 3600.</p>						
Token endpoint authorization	<p>Specifies whether the token endpoint accepts an existing session or requires credentials for authentication. The <code>pub.oauth:getToken</code> service functions as the token endpoint. Clients invoke this service to requests an access token from the Integration Server authorization server.</p> <table border="1"> <thead> <tr> <th>Select</th> <th>To</th> </tr> </thead> <tbody> <tr> <td>Accept existing session</td> <td>Indicate that the token endpoint service will accept requests from clients that have an active session on Integration Server. If these clients supply a valid session identified in the Cookie request header, the clients do not have to provide credentials to use the <code>pub.oauth:getToken</code> service. This is the default behavior and matches the behavior that existed prior to the Integration Server version 10.3.</td> </tr> <tr> <td>Require credentials</td> <td>Require clients to provide their credentials in the Authorization request header every time they request a new access token or refresh an existing access token by calling the <code>pub.oauth:getToken</code> service.</td> </tr> </tbody> </table> <p>Note: The <code>pub.oauth:getToken</code> service replaces the <code>pub.oauth:getAccessToken</code> and <code>pub.oauth:refreshAccessToken</code> services which are deprecated.</p> <p>Note: Token endpoint authorization impacts clients using the authorization code grant type, resource owner password credentials grant type, and client credentials grant type.</p>	Select	To	Accept existing session	Indicate that the token endpoint service will accept requests from clients that have an active session on Integration Server. If these clients supply a valid session identified in the Cookie request header, the clients do not have to provide credentials to use the <code>pub.oauth:getToken</code> service. This is the default behavior and matches the behavior that existed prior to the Integration Server version 10.3.	Require credentials	Require clients to provide their credentials in the Authorization request header every time they request a new access token or refresh an existing access token by calling the <code>pub.oauth:getToken</code> service.
Select	To						
Accept existing session	Indicate that the token endpoint service will accept requests from clients that have an active session on Integration Server. If these clients supply a valid session identified in the Cookie request header, the clients do not have to provide credentials to use the <code>pub.oauth:getToken</code> service. This is the default behavior and matches the behavior that existed prior to the Integration Server version 10.3.						
Require credentials	Require clients to provide their credentials in the Authorization request header every time they request a new access token or refresh an existing access token by calling the <code>pub.oauth:getToken</code> service.						

- Under Resource Server Settings, if you are configuring Integration Server as a resource server, in the **Authorization server** list, select the server that will be the authorization server.

You can use an Integration Server as the authorization server or you can use an external authorization server. The **Authorization server** list displays the configured remote server aliases and external authorization server aliases that are available for use.

If you intend to use a remote Integration Server as the authorization server and you have not already defined an alias for the authorization server, click the **Authorization server** link to go to the Settings > Remote Servers screen. For information about creating a remote server alias, see [“Setting Up Aliases for Remote Integration Servers” on page 115](#).

If you intend to use an external authorization server and you have not already defined an alias for the authorization server, click the **Add External Authorization Server** link to go to the External Authorization Server > Add screen. For information about creating an alias for an external authorization server, see [“Using an External Authorization Server” on page 630](#).

If you are configuring Integration Server as the authorization server only, Integration Server ignores the value of the **Authorization server** field.

6. Click **Save Changes**.

Defining Clients

Before a client application (client) can request access to a protected resource, you must use the Integration Server Administrator to register the client with the authorization server.

Registering Clients

Clients must be registered with the Integration Server acting as the authorization server to be able to access resources protected by OAuth. When you register a client, you identify the client as a confidential client or a public client, select the grant types the client can use, and specify token expiration and refresh information.

For each OAuth client registered with Integration Server, you must specify the OAuth grant types that the client can use to access the server. Keep the following information in mind when selecting the grant types that a client can use.

- Only confidential clients may use the client credentials grant and resource owner password credentials grants. You cannot select either of these grant types for public clients. If a client requests an access token using a grant that it is not allowed to use, Integration Server will reject the request and return an `unauthorized_client` OAuth error.
- When you register a new client, Integration Server automatically selects the Authorization Code Grant check box.
- The authorization code grant is the most secure of the four OAuth grant types supported by Integration Server.
- If you select the Authorization Code Grant or the Implicit Grant check boxes, you need to enter at least one Redirection URI for the client.
- After registering a client that will use the client credentials grant, you must give the client permission to access the resources it needs. This is different than the other grants. For the other

grants, Integration Server knows the owner because the resource owner is involved in the grant flow. Specifically, when an access token is being used to access a resource, Integration Server checks that the owner has permission to access the resource. With the client credentials grant, Integration Server does not know the owner. Consequently, it is the client who must have permission to access the resources. After registering the client, you must add the client to one or more user groups that belong to the access control lists (ACLs) that the client needs. The client ID generated for the client by Integration Server appears as the user name on the Security > User Management screens.

Note:

Prior to Integration Server version 10.3, registering a client did not require specifying allowed grant types. Rather, all registered clients were allowed to use the authorization code grant type and the implicit grant type. Beginning in Integration Server 10.3, all new clients must specify at least one grant type. Clients migrated to Integration Server 10.3 will still be able to use the authorization code grant type and the implicit grant types.

Complete the following steps to register a client with the authorization server.

➤ **To register a client**

1. Open Integration Server Administrator of the Integration Server defined as the authorization server if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Client Registration**.
4. Click **Register Client**.
5. Under **Client Configuration**, complete the fields as follows:

Field	Description
Name	<p>Specifies the name of the client.</p> <p>The name cannot contain the following characters:</p> <p>& () \ ; , / " : ' < ></p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: You cannot create clients with the same Name and Version combination.</p> </div>
Version	<p>Specifies the version number of the client.</p> <p>The version cannot contain the following characters:</p> <p>& () \ ; , / " : ' < ></p>

Field	Description						
	<p>Note: You cannot create clients with the same Name and Version combination.</p>						
Type	<p>Specifies the type of the client according to its ability to communicate with the authorization server.</p> <table border="1"> <thead> <tr> <th>Specify...</th> <th>When...</th> </tr> </thead> <tbody> <tr> <td>Confidential</td> <td>The client is capable of maintaining secure client authentications. When you select client type as Confidential, Integration Server generates a client secret. This client secret will be required by Integration Server when the client makes requests to the OAuth services. Specify Confidential when the OAuth session uses the authorization code grant type, resource owner password credentials grant type, or the client credentials grant type.</td> </tr> <tr> <td>Public</td> <td>The client is not capable of maintaining secure client authentications. Specify Public when the OAuth session uses the implicit grant type. For more information, see “Implicit Grant” on page 606.</td> </tr> </tbody> </table>	Specify...	When...	Confidential	The client is capable of maintaining secure client authentications. When you select client type as Confidential , Integration Server generates a client secret. This client secret will be required by Integration Server when the client makes requests to the OAuth services. Specify Confidential when the OAuth session uses the authorization code grant type, resource owner password credentials grant type, or the client credentials grant type.	Public	The client is not capable of maintaining secure client authentications. Specify Public when the OAuth session uses the implicit grant type. For more information, see “Implicit Grant” on page 606 .
Specify...	When...						
Confidential	The client is capable of maintaining secure client authentications. When you select client type as Confidential , Integration Server generates a client secret. This client secret will be required by Integration Server when the client makes requests to the OAuth services. Specify Confidential when the OAuth session uses the authorization code grant type, resource owner password credentials grant type, or the client credentials grant type.						
Public	The client is not capable of maintaining secure client authentications. Specify Public when the OAuth session uses the implicit grant type. For more information, see “Implicit Grant” on page 606 .						
Description	Optional. Specifies a description of the client.						
Redirect URIs	<p>Specifies the URIs that the authorization server will use to redirect the resource owner's browser during the grant process.</p> <p>You can add more than one URI at a time by specifying multiple lines, one URI to a line. Press the Enter key to separate lines.</p> <p>If you select the Authorization Code Grant or the Implicit Grant types, you need to enter at least one Redirection URI for the client.</p>						
Allowed Grants	<p>Specifies the OAuth grant types that the client can use to with Integration Server. Select one or more of the following:</p> <ul style="list-style-type: none"> ■ Authorization Code Grant ■ Implicit Grant ■ Client Credentials Grant 						

Field	Description
	<ul style="list-style-type: none"> Resource Owner Password Credentials Grant <p>The default is Authorization Code Grant which is the most secure grant type.</p> <p>If you select Authorization Code Grant or Implicit Grant, you need to enter at least one redirect URI for the client.</p> <p>Only confidential clients may use the client credentials and resource owner password credentials grants. You cannot select either of these grants for public clients. If a client requests an access token using a grant that it is not allowed to use, Integration Server will reject the request and return an <code>unauthorized_client</code> OAuth error.</p>

6. Under **Token**, specify the following information about the tokens issue to the client:

Field	Description								
Expiration Interval	<p>Specifies the length of time (in seconds) that the access token is valid.</p> <table> <thead> <tr> <th>Select...</th> <th>To...</th> </tr> </thead> <tbody> <tr> <td>Use OAuth Global Setting</td> <td>Use the setting specified in the Access token expiration interval field on the OAuth screen. This setting is shown in angle brackets. For more information about the Access token expiration interval field, see “Configuring OAuth Settings” on page 613.</td> </tr> <tr> <td>Never Expires</td> <td>Indicate that the access token never expires.</td> </tr> <tr> <td>Expires in</td> <td>Specify a specific time interval. Enter the number of seconds that the access token is valid in the field provided. The maximum value is 2147483647. The default is 3600.</td> </tr> </tbody> </table>	Select...	To...	Use OAuth Global Setting	Use the setting specified in the Access token expiration interval field on the OAuth screen. This setting is shown in angle brackets. For more information about the Access token expiration interval field, see “Configuring OAuth Settings” on page 613 .	Never Expires	Indicate that the access token never expires.	Expires in	Specify a specific time interval. Enter the number of seconds that the access token is valid in the field provided. The maximum value is 2147483647. The default is 3600.
Select...	To...								
Use OAuth Global Setting	Use the setting specified in the Access token expiration interval field on the OAuth screen. This setting is shown in angle brackets. For more information about the Access token expiration interval field, see “Configuring OAuth Settings” on page 613 .								
Never Expires	Indicate that the access token never expires.								
Expires in	Specify a specific time interval. Enter the number of seconds that the access token is valid in the field provided. The maximum value is 2147483647. The default is 3600.								
Refresh Count	<p>Specifies the number of times the access token can be refreshed.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: Tokens can only be refreshed when using the authorization code grant flow and the resource owner password credentials (ROPC) grant flow.</p> </div> <table> <thead> <tr> <th>Select...</th> <th>To...</th> </tr> </thead> <tbody> </tbody> </table>	Select...	To...						
Select...	To...								

Field	Description
Unlimited	Refresh the access token an unlimited number of times.
Limit	<p>Specify the number of times Integration Server can refresh the access token.</p> <p>If you specify any value greater than 0, Integration Server will issue a refresh token to enable the access token to be refreshed the specified number of times. When the access token expires, the client can use the <code>pub.oauth:getToken</code> service to submit a token refresh request to the authorization server.</p> <p>If you specify 0 or leave the Limit field empty, Integration Server does not issue a refresh token.</p> <p>The maximum value is 2147483647. The default is 0.</p>

7. Click **Save Changes**.

Integration Server generates a client ID. If you specified **Confidential** in the **Type** field, Integration Server also generates a client secret. Integration Server requires the client ID, client secret, or both when the client invokes the OAuth services.

Note:

When a confidential client is registered, a corresponding Integration Server user account is created. The user name is the client ID and the password is the client secret. If an existing client is changed from confidential to public or vice versa, the corresponding user account is created or deleted.

Note:

The password automatically generated for the Integration Server user account might not conform to password restrictions. However passwords that Integration Server generates automatically for OAuth clients are secure 36 character pseudo-random values.

Enabling and Disabling Clients

If you want to temporarily disable access to resources for all the access tokens issued to a registered client, you can disable that client. When you disable a client, the introspection endpoint (`pub.oauth:introspectToken`) returns "active=false" for all tokens issued to the client. This causes the resource server to deny access to requests that use one of the disabled client's tokens.

Note:

The `watt.server.oauth.disableClient.disableTokens` server configuration parameter controls whether or not the OAuth introspection endpoint, the `pub.oauth:introspectToken` service, considers whether a client account is disabled or enabled when determining if an access token is active. When `watt.server.oauth.disableClient.disableTokens` is set to `true`, the `pub.oauth:introspectToken` service considers the token to be inactive if the client account to which the token was issued is disabled. When set to `false`, the `pub.oauth:introspectToken` service does not consider the enabled/disabled state of the client account to which the access token was issued when evaluating an access token. For more information about `watt.server.oauth.disableClient.disableTokens`, see [“watt.server.” on page 1053/](#)

Complete the following steps to enable or disable a registered client.

➤ To enable or disable a client

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Client Registration**.
4. Under the **Active** column of the **Registered Clients** list, select one of the following:

Click	To
No	Enable a client.
Yes	Disable a client.

5. When prompted to confirm that you want to enable or disable the registered client, click **OK**.

Editing Clients

Complete the following steps to edit a registered client.

➤ To edit a client

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Client Registration**.
4. Under **Registered Clients**, click the **Application Name** or **Client ID** for the client that you want to edit.
5. Update the information for the client.

Note:

You cannot edit the data displayed in the **ID** or **Secret** columns.

6. Click **Save Changes**.

Deleting Clients

Complete the following steps to delete a registered client.

Important:

When you delete a client, Integration Server also deletes all the access and refresh tokens for the client.

> To delete a client

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Client Registration**.
4. Locate the client in the **Registered Clients** list, and click the **×** icon in the **Delete** column.
5. When prompted to confirm that you want to delete the registered client, click **OK**.

Defining Scopes

A scope indicates the resources the client can access on behalf of a resource owner. A scope consists of a name and one or more Integration Server folders, services, or both. If access is granted for a scope, then access is granted for all folders and services in that scope. You map the scope to a registered client to indicate the namespace resources the client can access on Integration Server. For more information about mapping scopes to clients, refer to “[Associating Scopes and Clients](#)” on [page 624](#).

When a request is made to the authorization server, Integration Server verifies that the scope is defined for a client. The client is allowed to access only the namespace resources that are specified for the scope. If the requested scope is not defined, Integration Server returns an error indicating that the scope is invalid.

Adding a Scope

Complete the following steps to add a scope. The scope defines the services and folders that the clients are authorized to access.

> To add a scope

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Scope Management**.
4. Click **Add Scope**.
5. Under **Scope Configuration**, specify the following information:

Field	Description
Name	<p>Specifies a unique name for the scope. The scope name must consist of ASCII character codes in the range of 33 - 126, must be 100 characters or less, and cannot contain any of the following characters:</p> <p style="margin-left: 20px;">\ , "</p>
Description	<p>A description of the scope.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: If Integration Server generates the scopes automatically while creating the REST API descriptor based on the Swagger document, then IS adds a default scope description as XX. You can modify this description; however, when IS re-generates the RAD using the same Swagger document, IS overrides the description as XX</p> </div>
Folders and services	<p>Specifies the list of folders and individual services that the client can access on behalf of the resource owner.</p> <p>You can add more than one folder or service at a time by specifying multiple lines, one folder or service to a line. Press Enter to separate lines.</p>
URL templates	<p>Specifies a URL template for a REST V2 resource that you want to secure using the scope. The format of URL template is <i><HTTP method>_<REST API descriptor namespace>/<URL template of resource></i>. For example,</p> <p style="margin-left: 20px;">GET_RadNamesSpace/employee.</p>

6. Click **Save Changes**.

Editing Scopes

Complete the following steps to edit a scope.

➤ **To edit a scope**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Scope Management**.
4. Under **Scope Configuration**, click the **Name** for the scope that you want to edit.
5. Update the information for the scope.

Note:

You cannot change the **Name** field for a scope.

6. Click **Save Changes**.

Deleting Scopes

Complete the following steps to delete a scope.

Note:

You cannot delete a scope that is used by a client. To see whether a scope is in use by a client, see [“Viewing Associations Between Clients and Scopes”](#) on page 627.

➤ **To delete a scope**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Scope Management**.
4. Locate the scope in the **Defined Scopes** list, and click the **×** icon in the **Delete** column.
5. When prompted to confirm that you want to delete the scope, click **OK**.

Associating Scopes and Clients

You manage the scope and client associations on the Associate Scopes to Clients page.

The Associate Scopes to Clients page is divided into two areas:

- You can associate scopes to clients in the **Scopes** area (on the left). Use the **Scopes** area to make associations when you want to associate multiple clients to a single scope. This offers the advantage of adding or removing several associated clients for the scope at one time.
- You can associate clients to scopes in the **Clients** area (on the right). Use the **Clients** area to make associations when you want to associate multiple scopes to a single client. This offers the advantage of adding or removing several associated scopes for the client at one time.

Note:

You must define scopes and clients *before* you can associate them. For more information about registering clients, see [“Defining Clients” on page 616](#). For more information about adding scopes, see [“Defining Scopes” on page 622](#).

Adding Associations Between Clients and Scopes

Use the following procedure to add associations between scopes and clients.


➤ To add associations between scopes and clients

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Scope Management**.
4. Click **Associate Scopes to Clients**.
5. If you want to associate one or more clients to a particular scope, do the following:
 - a. Under **Scopes**, in the **Select Scope** list, select the scope to which you want to add an association to one or more clients.

The **Scopes** area of the Associate Scopes to Clients screen (on the left) contains two lists. **Clients associated with Scope** is a list of clients currently associated to the selected scope. **Remaining Clients** is a list of clients not currently in the selected scope.

- b. In the **Remaining Clients** list, select (highlight) the client or clients that you want to add to the scope.

To select additional clients without deselecting currently selected clients, press the Ctrl key while you click the clients you want to select. To deselect a client, press Ctrl while you click the currently selected entry.

- c. After you have selected all the clients you want to add to the scope, click . Integration Server moves the selected clients to the **Clients associated with Scope** list.


6. If you want to associate one or more scopes to a particular client, do the following:

- a. Under **Clients**, in the **Select Client** list, select the client to which you want to add an association to one or more scopes.

The **Clients** area of the Associate Scopes to Clients screen (on the right) contains two lists. **Scopes in this Client** is a list of scopes currently associated to the selected client. **Remaining Scopes** is a list of clients not currently in the selected scope.

- b. In the **Remaining Scopes** list, select (highlight) the scope or scopes that you want to add to the client.

To select additional scopes without deselecting currently selected scopes, press Ctrl while you click the scopes you want to select. To deselect a scope, press Ctrl while you click the currently selected entry.

- c. After you have selected all the scopes you want to add to the client, click . Integration Server moves the selected scopes to the **Scopes in this Client** list.

7. Click **Save Changes**.

Removing Client and Scope Associations


Use the following procedure to remove client and scope associations.


Note:

Removing associations between scopes and clients does not delete the clients or scopes from the system, nor does it affect access tokens that have already been issued. For more information about deleting clients, see [“Deleting Clients” on page 622](#). For more information about deleting scopes, see [“Deleting Scopes” on page 624](#).

➤ To remove client and scope associations

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Scope Management**.
4. Click **Associate Scopes to Clients**.
5. If you want to remove an association of one or more clients from a particular scope, do the following:
 - a. Under **Scopes**, in the **Select Scope** list, select the scope for which you want to remove an association from one or more clients.
 - b. In the **Clients associated with Scope** list, select (highlight) the client or clients that you want to remove from the scope.

- To select additional clients without deselecting currently selected clients, press Ctrl while you click the clients you want to select. To deselect a client, press Ctrl while you click the currently selected entry.
- c. After you have selected all the clients you want to remove from the scope, click . Integration Server moves the selected clients to the **Remaining Clients** list.
6. If you want to remove an association of one or more scopes from a particular client, do the following:
 - a. Under **Clients**, in the **Select Client** list, select the client for which you want to remove an association from one or more scopes.
 - b. In the **Scopes in this Client** list, select (highlight) the scope or scopes that you want to remove from the client.

To select additional scopes without deselecting currently selected scopes, press Ctrl while you click the scopes you want to select. To deselect a scope, press Ctrl while you click the currently selected entry.
 - c. After you have selected all the scopes you want to remove from the client, click . Integration Server moves the selected scopes to the **Remaining Scopes** list.
 7. Click **Save Changes**.

Viewing Associations Between Clients and Scopes

Use the following procedure to view associations between scopes and clients.

➤ To view the associations between clients and scopes

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Scope Management**.
4. Click **Associate Scopes to Clients**.
5. Do one of the following:
 - To view clients associated with a particular scope, under **Scopes**, in the **Select Scope** list, select the scope for which you want to view associated clients.
 - To view scopes associated with a particular client, under **Clients**, in the **Select Client** list, select the client for which you want to view associated scopes.

Managing Tokens

Use the Tokens screen to view and remove the active tokens issued by the authorization server. Client applications use these tokens to access the resources on Integration Server (the resource server). When you delete the tokens, the client application can no longer access the resources owned by the resource owners.

You can also use built-in services include with the WmPublic package to revoke tokens issued by an Integration Server acting as an authorization server and remove expired access tokens.

Viewing Tokens

Use the following procedure to view the active tokens issued by the authorization server.

➤ To view tokens

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Tokens**. Under **Tokens**, the Integration Server Administrator lists the active and expired access tokens issued by Integration Server.

Deleting Tokens

You can use the Tokens screen to delete all active tokens. When you delete the token from the list of active tokens, Integration Server deletes both the access token and the refresh token. Keep the following points in mind when deleting tokens:

- After you delete a token, other resource owners using the same client application can continue to do so. To keep a particular client application from accessing resources, you can disable or delete the client application. For more information about deleting client applications, see [“Deleting Clients” on page 622](#).
- Once a token is deleted, Integration Server rejects any requests from clients using that access or refresh token.

➤ To delete tokens

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click **Tokens**.
4. Locate the token in the **Tokens** list and click the **×** icon in the **Delete** column.

- When prompted to confirm that you want to delete the token, click **OK**.

Revoking Tokens

You can revoke a token issued by Integration Server using the `pub.oauth:revokeToken` service in the `WmPublic` package. The `pub.oauth:revokeToken` service revokes all of the tokens related to a specified token. If the token is an access token, the service revokes the associated refresh token as well. If the token is a refresh token, the service revokes the associated access token as well.

For more information about the `pub.oauth:revokeToken` service, see *webMethods Integration Server Built-In Services Reference*.

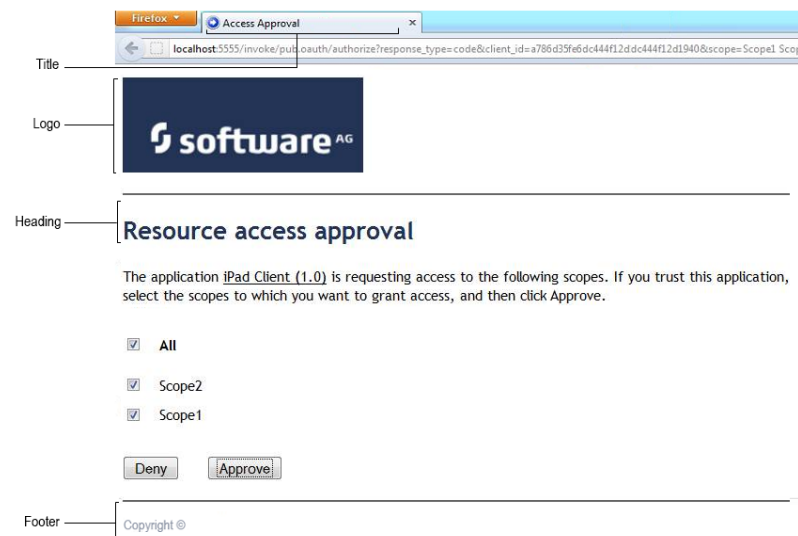
Removing Expired Access Tokens

You can use the `pub.oauth:removeExpiredAccessTokens` service in the `WmPublic` package to remove expired OAuth access tokens from the database. For more information about the `pub.oauth:removeExpiredAccessTokens` service, see *webMethods Integration Server Built-In Services Reference*.

Customizing the Approval Page

The approval page is an HTML page the authorization server sends to the resource owner after a client submits a request for access to their private resources. The resource owner uses the page to accept or deny the request.

OAuth approval default page



You can customize the title, logo, heading, and footer of the approval page using the `watt.server.oauth.approvalpage` properties. For information about using these properties, see [“Server Configuration Parameters” on page 1017](#).

About Using Integration Server as the Resource Server

When a client has requested access to a protected resource and has received an access token, the client can then present the access token to the resource server. If Integration Server is the resource server, clients can present the access token using the authorization request header field. Clients can send the access token in the Authorization request header field using the Bearer authentication scheme to submit the access token. For example:

```
GET /invoke/folder/svc HTTP/1.1
Host: your-company.com:5555
Authorization: Bearer access_token_id
```

If you are using the `pub.client:http` service, you can send the token using the `auth` header field. For information about using this service, see *webMethods Integration Server Built-In Services Reference*.

For more information about the Bearer authentication scheme, refer to the OAuth 2.0 Authorization Framework Bearer Token Usage specification.

Using an External Authorization Server

When Integration Server is the resource server, you must specify an authorization server. As an alternative to using an Integration Server as the authorization server, you can use a third party server as the authorization server. This allows Integration Server to use OAuth bearer tokens created by a third party OAuth 2.0 authorization server where the third-party vendor supports RFC 7662, OAuth 2.0 Token Introspection.

To use an external authorization server, you must:

- Configure your third-party authorization server. This includes, but is not limited to, the following.
 - Create a client account that Integration Server will use to call the authorization server's introspection endpoint.

Make a note of the `client_id` and `client_secret` values. You will provide this information as part of defining the external authorization server alias for the Integration Server resource server.

Make a note of the URL for the introspection endpoint. You will provide this information as part of defining the external authorization server alias in the Integration Server resource server.
 - Create one or more OAuth scopes. These must match the names of the OAuth scopes you create in the Integration Server resource server.

For more information on creating and configuring an OAuth 2.0 authorization server, consult the documentation provided by the vendor.

- Configure an alias to the authorization server. For information, see [“Creating an External Authorization Server Alias” on page 631](#).

- Select the external authorization server alias as the **Authorization Server** value on the **Security > OAuth > Edit Global Settings** page.

Currently, Integration Server can be used with an external authorization server that supports RFC 7662, OAuth 2.0 Token Introspection, including:

- Okta
- Ping Identity


Creating an External Authorization Server Alias

If you want an Integration Server functioning as a resource server to use a third party server as the authorization server, you must create an external authorization server alias.

➤ To create an external authorization server alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. Click Add External Authorization Server.
4. Under External Authorization Server Settings, provide the following information.

Field	Description
Name	Alias for the external authorization server. The following characters are prohibited: ? [] / \ = + < > : ; " , * ^ @
Introspection Endpoint	The URL of the introspection endpoint for the external authorization server. Integration Server uses the introspection endpoint to determine if access tokens used in client requests are currently active.
Client Id	The ID of the user account that Integration Server uses when sending requests to the introspection endpoint of the external authorization server.
Client Secret	The password for the user account that Integration Server uses when sending requests to the introspection endpoint of the external authorization server.
User	The Integration Server user account that Integration Server uses to execute the client request. If the client is requesting a service, this is the user account that Integration Server uses to execute the service, which occurs after Integration Server calls the introspection endpoint. If the client is requesting a file, this is the user account that Integration Server uses to access the file.

Field	Description
	<p>The User value is used only if the introspection endpoint of the external authorization server indicates that the access token is currently active.</p> <p>Click  to search for and select your user. A user can be selected from the local or central directory.</p>
Keystore Alias (optional)	<p>The alias of the keystore on Integration Server that holds the digital certificate that Integration Server sends to the external authorization server during the mutual (two-way) SSL handshake. You need to select a keystore alias only when the client account on the external authorization server is configured to use mutual (two-way) SSL.</p>
Key Alias (Optional)	<p>The alias of the Integration Server private key and associated digital certificate that Integration Server sends to the external authorization server during the mutual (two-way) SSL handshake. You need to select a key alias only when the client account on the external authorization server is configured to use mutual (two-way) SSL.</p>
Truststore Alias (Optional)	<p>The alias of the truststore on Integration Server that holds the Certificate Authority (CA) certificate of the external authorization server. You need to select a truststore alias only when all of the following are true:</p> <ul style="list-style-type: none"> <li data-bbox="386 1045 1166 1108">■ The client account on the external authorization server is configured to use mutual (two-way) SSL, and <li data-bbox="386 1140 1227 1241">■ The authorization server's Certificate Authority certificate is not in the set of well-known authorities trusted by the JVM in which Integration Server runs, and <li data-bbox="386 1272 1235 1329">■ The <code>watt.security.cert.wmChainVerifier.trustByDefault</code> property is set to false.
Default Scope	<p>Default scope that takes effect when no scope is explicitly stated in the response from the external authorization server's introspection endpoint. Enter the scope values in this field exactly as they are defined on the external authorization server.</p> <p>When responses from an external authorization server's introspection endpoint do not return a scope value and a Default Scope is not specified, Integration Server considers requests bearing the access token from the authorization server to be out of scope and rejects the requests with a 401 response.</p>

5. Click **Save**.

Integration Server Administrator displays the new external authorization server alias under External Authorization Servers on the Security > OAuth page.

Deleting an External Authorization Server

If you no longer need to use a third party server as the authorization server for an Integration Server acting as the resource server, you can delete the external authorization server alias.

➤ To delete an external authorization server alias

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigational Panel, click **OAuth**.
3. In the External Authorization Servers list, locate the external alias in and click the **×** icon in the **Delete** column.
4. When prompted to confirm that you want to delete the alias, click **OK**.

34 Configuring a Central User Directory or LDAP

■ Before You Begin	636
■ Overview of How Integration Server Works with Externally Defined Users and Groups .	636
■ Configuring Central User Management	638
■ Overview of Using LDAP	640
■ Configuring the Server to Use LDAP	641
■ Considerations for User Accounts and Groups	647
■ About Granting Administrator Privileges to External Users	649
■ Granting Developer Privileges to External Users	650
■ Granting Access to Services and Files to External Users	651

Before You Begin

This chapter describes how Integration Server uses an external directory to authenticate clients, rather than using internally defined user and group information. For information about using internally defined users and groups, refer to [“Managing Users and Groups” on page 83](#). You can set up Integration Server to access information from an external directory if your site uses one of the following external directories for user and group information:

- Central user management via Common Directory Services and My webMethods Server
- Lightweight Directory Access Protocol (LDAP)

You can configure Integration Server to use only one external directory at a time, a central user directory or LDAP.

Important:

If you are using My webMethods Server in your environment, Software AG recommends that you configure Integration Server to work with a central user management directory. If you want to use an LDAP server and you are using My webMethods Server, it is recommended that you configure LDAP using My webMethods Server. Configure Integration Server to work with the LDAP server directly only when you are not using My webMethods Server.

Before you continue reading this chapter, you may find it helpful to first understand how Integration Server uses user and group information. Read the following sections if you have not already done so.

- [“Managing Users and Groups” on page 83](#)
- [“Adding an Administrator User” on page 88](#)
- [“Adding a Developer User” on page 89](#)
- [“Controlling Access to Resources with ACLs” on page 501](#)
- [“Basic Authentication” on page 516](#)

Overview of How Integration Server Works with Externally Defined Users and Groups

The following sections provide information about how and when Integration Server interacts with users and groups defined in a central user directory or in LDAP, specifically:

- How externally defined users and groups can be used in Integration Server.
- When Integration Server accesses information about externally defined users and groups.
- How Integration Server authenticates users who belong to externally defined groups or roles.

How the Server Uses Externally Defined Users and Groups

Integration Server can use externally defined information for the same purposes it uses internally-defined user and group information:

- To authenticate clients using user names and passwords
- To control who can configure and manage Integration Server
- To control who can create, modify, and delete services using Software AG Designer
- To control access to services and files that are available in Integration Server

Externally defined information does not replace ACLs. To control access to services and files, you still need to set up the ACLs that identify the groups that are allowed and denied access to specific services and files. However, you can assign externally defined groups to an ACL.

When you configure the server to use central user management or LDAP directory, externally defined users and groups are not displayed on the **Security > User Management** page. However, if an external group has been mapped to an Integration Server ACL, the group will be displayed on the **Security > Access Control Lists** page.

Note:

If the name of an externally defined user group contains an apostrophe ('), Integration Server will not display the group on the **Security > Access Control Lists** page.

When the Server Accesses Externally Defined Information

Integration Server obtains externally defined information to authenticate clients and to determine whether an ACL allows or denies an action.

Note:

Client requests that require Integration Server to access a central user directory or an LDAP directory may take longer to complete than those that do not.

How Integration Server Authenticates Externally Defined Clients

When authenticating clients using user names and passwords, Integration Server always looks for the user account internally before looking in an external directory, specifically:

1. When Integration Server finds an internally-defined user account for the supplied user name, Integration Server authenticates the client using the internally-defined information. If the supplied user name and password combination is correct, authentication succeeds and Integration Server proceeds with the request. If authentication fails and an external directory is not configured, Integration Server rejects the request with an "Invalid credentials" error.
2. If authentication fails and an external directory is configured (either a central user directory or LDAP), Integration Server requests that the external directory authenticate the client. If authentication succeeds, Integration Server proceeds with the request. If authentication fails, Integration Server rejects the request with an "Invalid credentials" error.

For example, if a user account is defined in the My webMethods Server user directory, Integration Server authenticates the client using the information defined in the My webMethods Server database. If the supplied password is correct, Integration Server proceeds with the request. If the supplied password is not correct, Integration Server rejects the request.

Note:

If the passwords are contained in an external authentication system other than Central Users or LDAP, you must create your own pluggable module to obtain this information. See [“Customizing Authentication Using JAAS” on page 527](#) for information about setting up a pluggable module.

3. If Integration Server cannot find either an internally or externally defined user account for the user, Integration Server rejects the request.

If the user does not supply a user name or password, the server uses the internally-defined Default user account. This account grants access to resources that allow anonymous access.

Note:

A local user can have the same name as a user in an external user directory. If the locally defined user and the externally defined user have the same password and the supplied password is correct, Integration Server authenticates the user with the privileges defined locally. This occurs because Integration Server checks its local user list first. If the passwords are different for the locally and externally defined user accounts, Integration Server treats the user accounts as two different users. An authentication request for the user that includes the externally defined password results in the external directory authenticating the user and granting the user the externally defined privileges.

Note:

The ability to have a locally defined user and an externally defined user with the same name is available after applying a fix that includes PIE-62998 (IS_10.5_Core_Fix5 and higher).

Configuring Central User Management

Central user management involves using a single location to store and manage information about users of webMethods products. You can use Integration Server Administrator to grant users in a central directory access to Integration Server functionality and services. For example, you can assign a My webMethods Server role or group to an ACL.

If a user will access Integration Server or Trading Networks through My webMethods interfaces, create the users in My webMethods Server and then use Integration Server Administrator to give them access to the necessary areas. If such users are already defined in an external directory such as LDAP, you can configure My webMethods Server to work with the external directory. When configured this way, authentication and authorization requests are still made by the Integration Server; however, the server creates the connection to the external directory using the directory settings defined by the My webMethods Server database.

Users defined in a central location, such as the My webMethods Server user directory, are sometimes referred to as central users. The ability to use a central user directory for authentication is provided by the common directory services that are part of My webMethods Server.

Requirements for Integration Server to Use Central User Management

- My webMethods Server must be installed and configured to use an external database.
- My webMethods Server clustering must be configured even if Integration Server connects to a standalone My webMethods Server. This is because the client-side common directory services APIs used by Integration Server to communicate with My webMethods Server functions as a cluster node. My webMethods Server must be started at least once to enable My webMethods Server clustering.

Note:

Beginning with My webMethods Server 10.0, My webMethods Server must use a Universal Messaging server as its JMS provider. My webMethods Server uses a JMS provider for cluster communication and synchronization.

- Integration Server must have a JDBC connection pool that points to the My webMethods Server database component, and the CentralUsers functional alias must point to that connection pool. On the **Settings > Resources** screen, **MWS SAML Resolver URL** field must point to your My webMethods Server host and port.
- The Single Sign On with My webMethods Server must be configured to point to the My webMethods Server host and port. For more information about this property, see [“Configuring the MWS Single Sign-On Resource Setting” on page 526](#).

Note:

When Integration Server is successfully configured to use central user management, and the logging level for the server log facility 0024 User Manager is set to at least Info, Integration Server logs the message [ISS.0024.0012I] Central User Management initialized successfully upon startup.

If you later want to stop using central user management, follow the instructions in [“Connecting Integration Server to Database Components in an External RDBMS” on page 139](#), but in the **Associated Pool Alias** list, click **None** for the **CentralUsers** function, and then restart Integration Server.

Note: Integration Server updates the Anonymous ACL automatically to include the My webMethods Users Role from My webMethods Server.

For more information about configuring My webMethods Server for common directory services and clustering, see *Administering My webMethods Server*.

Note:

If Central User Management is configured to use LDAP over SSL, set the `javax.net.ssl` properties in the `custom_wrapper.conf` file.

Considerations for My webMethods Server Query Roles

Integration Server can slow or terminate unexpectedly when handling service invocations if Integration Server has to evaluate all roles defined in My webMethods Server, including LDAP query and database query roles. You might encounter this issue when invoking Integration Server services using WS Client Connectors from a CAF application, especially if the following conditions are true:

- When there are LDAP query and database query roles defined in My webMethods Server.
- Common Users is enabled in Integration Server.
- The services have ACLs assigned to them.

Evaluation of these roles depend on external query execution and might impact Integration Server performance.

You can control whether Integration Server evaluates LDAP or database query roles in the server using the `watt.cds.skip.role.types` extended setting. If LDAP or database query roles are not used for any ACL management in Integration Server with Central Users enabled, you might consider disabling the query roles evaluation function.

For more information about using the `watt.cds.skip.role.types` extended setting, see [“Server Configuration Parameters” on page 1017](#).

Overview of Using LDAP

If your site uses Lightweight Directory Access Protocol (LDAP) for user and group information, you can configure the Integration Server to obtain user and group information from the external directory. You can configure Integration Server to use more than one LDAP directory at a time, allowing Integration Server to work with different LDAP directories for users in different locations or different organizations. In addition, you can maintain multiple LDAP directories so that one directory serves as a backup for another.

Important:

If you want to use an LDAP server to store user information and you are using My webMethods Server, it is recommended that you configure LDAP using My webMethods Server. Configure Integration Server to work with LDAP server directly only when you are not using My webMethods Server.

LDAP protocols are designed to facilitate sharing information about resources on a network. Typically, they are used to store profile information (login ID, password, etc.). You can also use them to store additional information. Integration Server uses LDAP for performing external authentication.

Using your existing LDAP information allows you to take advantage of a central repository of user and group information. System administrators can add and remove users from the central location. Users do not need to remember a separate password for webMethods applications; they can use the same user names and passwords that they use for other applications. Remember to use your LDAP tools to administer users or groups stored in an external directory.

About LDAP and Caching

For LDAP, after accessing user information, the Integration Server caches it to improve performance. If the information remains in the cache for one hour without being accessed, or if the cache space is needed for a more recent request, the Integration Server deletes the information from the cache.

If the server receives subsequent requests that require the information it has in cache, the Integration Server uses the cached information rather than accessing the external directory.

Configuring the Server to Use LDAP

To configure the server to use LDAP, you need to:

- Instruct Integration Server to use the LDAP protocol
- Define one or more configured LDAP servers that the Integration Server is to use for these users
- If an LDAP provider is SSL-enabled, you can set the `watt.server.ssl.trustStoreAlias` property to point to the truststore alias that contains the certificates required to establish a secure connection with the LDAP server.

Software AG recommends that you use central user management instead of configuring Integration Server to use one more LDAP directories for external user management. For more information about central user management, see [“Configuring Central User Management” on page 638](#) and *Administering My webMethods Server*.

➤ To specify LDAP as the external provider

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **LDAP Configuration**.
4. Click **Edit LDAP Configuration**.
5. Next to **Provider**, select **LDAP**.

Integration Server issues a prompt to verify that you want to change the setting. Click **OK**. If your Integration Server is configured to central user management, you must disable it before you can configure LDAP. For information about disabling central user management, see [“Configuring Central User Management” on page 638](#).

6. Enter the following information:

For this field...	Specify...
Cache Size (Number of Users)	<p>The maximum number of LDAP users Integration Server can keep in memory in the user cache. The default is 10.</p> <p>Once the limit is reached, Integration Server selects users for removal from the cache based on how long they have been idle. As a result, activity can extend the time a user remains in the cache.</p> <p>As a general rule, specify a cache size equivalent to 5-10% of the number of users in your LDAP system. However, if only a few sessions are ever logged on simultaneously, set the cache size to be the same as the number of simultaneous sessions.</p>
Credential Time-to-Live (Minutes)	<p>The number of minutes an LDAP user's credentials (userid and password) can remain in the credential cache before being purged. The default is 60 minutes.</p> <p>When a user first attempts to log in, Integration Server creates a user object and checks the user's credentials against the LDAP directory. Integration Server stores the credentials so that subsequent requests to authenticate will be made against the cached credentials, not the LDAP directory.</p> <p>For security reasons, you can control the length of time these cached credentials are valid. The credentials are secure because they are stored using a one-way hashing function, and cannot be recovered from the cache. If a user attempts to log in with credentials that do not match the cached version, Integration Server flushes the cache and checks the credentials against the LDAP directory. If the credentials are valid, the Integration Server caches them; otherwise, the cache remains empty.</p> <p>For normal secure environments, a time-to-live value between one hour and one day is adequate. For higher security environments, a time-to-live of between one and five minutes may be more appropriate.</p> <p>The Time-to-Live is absolute; therefore, activity will not cause the credentials to remain in cache longer.</p>

7. Click **Save Configuration**.

To finish configuring Integration Server to use an LDAP directory, continue to the procedure [“Defining an LDAP Directory to Integration Server ” on page 643](#).

Defining an LDAP Directory to Integration Server

➤ To define an LDAP directory to Integration Server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **LDAP Configuration**.
4. Click **Add LDAP Directory**.
5. On the **Settings > LDAP Directory > Add** screen, enter the following information:

For this parameter	Specify...
Directory URL	<p>The complete URL of the LDAP server. The URL has the format <i>protocol://hostname:portnumber</i> where</p> <ul style="list-style-type: none"> ■ The <i>protocol</i> is LDAP for standard connections or LDAPS for secure connections. ■ The <i>host</i> is the host name or IP address of the LDAP server. The <i>port</i> is the port on which the server is running. The port is optional. If omitted, the port defaults to 389 for LDAP, or 636 for LDAPS. <p>For example, specifying the URL <code>ldaps://ldapserv1:700</code> would create a secure connection to the LDAP server running on the non-standard port 700 on the host called <code>ldapserv1</code>.</p> <p>If you specify <code>ldaps</code>, Integration Server attempts to make a secure connection to the directory server using an SSL socket. If the directory server is configured to use SSL, it will have a server certificate in place to identify itself to clients. This certificate must be signed by an authority to prove its validity (i.e. the server certificate is signed by a CA). By default, the Integration Server will only trust certificates signed by a signing authority whose CA certificate is in the Integration Server's trusted CAs directory. Refer to “Configuring Integration Server for Secure Communication” on page 457 for instructions on configuring the trusted CAs directory and finding the CA certificate.</p>
Principal	<p>The user ID the Integration Server should supply to connect to the LDAP server, for example, <code>o=webm.com</code> or <code>dc=webm,dc=com</code>.</p> <p>This user should not be the Administrator account, but a user that has permission to query groups and group membership. If</p>

For this parameter	Specify...
Credentials	<p>your LDAP server allows anonymous access, leave this field blank.</p> <p>The password the Integration Server should supply to connect to the LDAP server, that is, the Principal's password. The Integration Server encrypts this password according to the settings specified on the Outbound Passwords screen. For more information, see “Configuring Integration Server for Secure Communication” on page 457.</p>
Connection Timeout (seconds)	<p>The number of seconds the Integration Server will wait while trying to connect to the LDAP server. After this time has passed, the Integration Server will try the next configured LDAP server on the list. The default is 5 seconds. Increase this number if your network has latency problems. If most requests will be from batch processes, you can increase this number to be 30 seconds or more.</p>
Minimum Connection Pool Size	<p>The minimum number of connections allowed in the pool that the Integration Server maintains for connecting to the LDAP server. When the Integration Server starts, the connection pool initially contains this minimum number of connections. The Integration Server adds connections to the pool as needed until it reaches the maximum allowed, which is specified in the Maximum Connection Pool field. The default is 0.</p>
Maximum Connection Pool Size	<p>The maximum number of connections allowed in the pool that the Integration Server maintains for connecting to the LDAP server. When the Integration Server starts, the connection pool initially contains a minimum number of connections, which are specified in the Minimum Connection Pool field. The Integration Server adds connections to the pool as needed until it reaches the maximum allowed. The default is 10.</p>
Synthesize DN	<p>Builds a distinguished name by adding a prefix and suffix to the user name.</p> <p>The Synthesize DN method can be faster than the Query DN method (see below) because it does not perform a query against the LDAP directory. However, if your LDAP system does not contain all users in a single flat structure, use the Query DN method instead.</p> <p>DN Prefix</p> <p>A string that specifies the beginning of a DN you want to pass to the LDAP server.</p> <p>DN Suffix</p>

For this parameter	Specify...
	<p>A string that specifies the end of a DN you want to pass to the LDAP server.</p> <p>For example, if the prefix is "cn=" and the suffix is ",ou=Users" and a user logs in specifying "bob", the Integration Server builds the DN <code>cn=bob,ou=Users</code> and sends it to the LDAP server for authentication.</p> <div data-bbox="662 506 1461 751" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Be sure to specify all the characters required to form a proper DN. For instance, if you omit the comma from the suffix above, that is, you specify "ou=Users" instead of ",ou=Users", the Integration Server will build the invalid DN (<code>cn=bobou=Users</code>).</p> </div>
<p>Query DN</p>	<p>Builds a query that searches a specified root directory for the user.</p> <p>Use this method instead of the Synthesize DN method (see above) if your LDAP directory has a complex structure.</p> <p>UID Property</p> <p>A property that identifies an LDAP userid, such as "cn" or "uid".</p> <p>User Root DN</p> <p>Enter the full distinguished name. For example, if you specify <code>ou=users,dc=webMethods,dc=com</code>, the Integration Server will issue a query that starts searching in the root directory <code>ou=users</code> for a common name that matches the name the user logged in with.</p>
<p>User Email</p>	<p>Specifies the name of the attribute that is used to store users' email addresses. For example, "mail".</p>
<p>Default Group</p>	<p>An Integration Server group with which the user is associated. The user is allowed to access services that members of this Integration Server group can access. This access is controlled by the ACLs with which the group is associated.</p> <p>If you also specify a value in the Group Member Attribute field, the user has the same access as members of the Integration Server group <i>and</i> members of LDAP groups that have been mapped to an Integration Server ACL.</p> <div data-bbox="662 1738 1461 1869" style="background-color: #f0f0f0; padding: 5px;"> <p>Important: Do not specify Anonymous as the default group if any user in this group needs to have administrator privileges. The</p> </div>

For this parameter	Specify...
	<p>default ACL denies the Anonymous group and will not allow access the root page. Choose the appropriate group in the Default Group field to ensure that the required ACLs get assigned to your group.</p> <p>Important: You must specify a value in the Group Member Attribute field, the Default Group field, or both.</p>
Group Member Attribute	<p>The name of the attribute in a group's directory entry that identifies each member of the group. This value is usually "member" or "uniqueMember", but can vary depending on the schema of the LDAP directory.</p> <p>Integration Server uses this information during ACL checking to see if the user attempting to log in belongs to an LDAP group that has been mapped to an ACL.</p> <p>If no value is specified here, Integration Server does not check for membership in an LDAP group. As a result, the user's ability to access Integration Server services is controlled by the Integration Server group specified in the Default Group field.</p> <p>Note: You must specify a value in the Group Member Attribute field, the Default Group field, or both.</p>
Group ID Property	A property that identifies an LDAP group, such as CN.
Group Root DN	<p>The full distinguished name.</p> <p>For example, if you specify <code>ou=groups,webMethods,dc=com</code>, Integration Server will issue a query that will display all the LDAP groups.</p> <p>Note: You must specify values in the Group ID Property and Group Root DN fields.</p>

6. Click **Save Changes**.

The LDAP Directory List displays the added the LDAP directory.

7. Click **Move Up/Move Down** to order the directories in the list based on their priority.

Note:

If you define multiple LDAP servers, Integration Server will search the LDAP directories in the order in which they are displayed on the **Security > User Management > LDAP**

Configuration screen. If Integration Server does not find the user in the first LDAP directory, it will search in order through the list.

Mapping an LDAP User's Access to ACLs

As with Integration Server groups, you can associate LDAP groups with ACLs to control access to Integration Server resources. Associating an LDAP group with an ACL is referred to as *mapping*. ACL mapping to LDAP groups can be done directly through the **Security > ACLs** page. For more information about allowing groups access to ACLs, refer to [“Allowing or Denying Group Access to ACLs” on page 506](#).

Stopping Use of an LDAP as an External Directory

If you no longer want to use LDAP as an external directory, you can update the configuration to remove the external directory configuration settings.

➤ To stop using a LDAP as an external directory

1. Open the Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **User Management**.
3. Click **Edit LDAP Configuration**.
4. Click **Local** in the **Provider** field.
5. Click **Save Configuration**.
6. Click **OK**.
7. Restart your Integration Server.

Considerations for User Accounts and Groups

This section provides information about user accounts and groups that you should consider if you are using an external directory for user and group information.

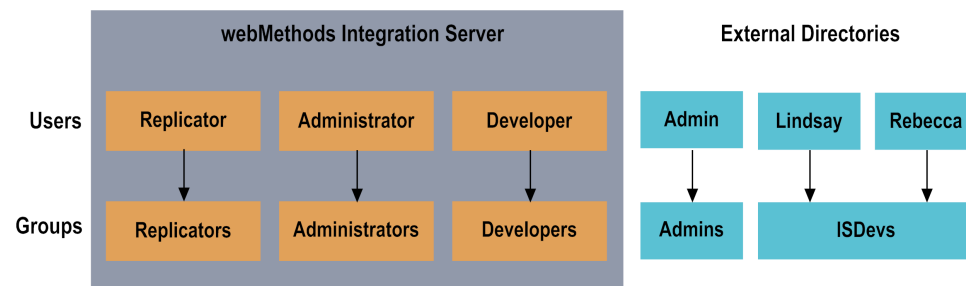
- **You should keep internal and external user accounts and group names unique.** It might get confusing if you have an external user account that has the same user name as an internal user account or an external group with the same group name as an internal group. For more information, see [“About Keeping Internal and External User Accounts and Group Names Unique” on page 648](#).
- **You cannot use the Integration Server Administrator to manage (i.e., create, edit, or delete) Central Users.** You must use My webMethods Server to administer Central Users and Directories. Refer to *Administering My webMethods Server* for more information.

- **You cannot use the Integration Server Administrator to manage (i.e., create, edit, or delete) LDAP user and group information.** To make changes to LDAP directories, follow your site's standard directory update procedures.
- **You can perform package replication without using the predefined Replicator account.** You can use a different account for package replication as long as the subscription requester specifies an account that is a member of a group that is assigned to the Replicators ACL. For more information, see [“About User Groups and Package Replication” on page 648](#).

About Keeping Internal and External User Accounts and Group Names Unique

Software AG recommends that you keep user names and group names unique between internal and external user accounts and groups. Having an external user account that has the same user name as an internal user account, or an external group with the same group name as an internal group might get confusing. If you do have identically named user names or group names, the server always uses the internally-defined information.

To avoid confusion, Software AG recommends that you do not set up user accounts or groups internally if you are using an external directory. The exceptions are the predefined user accounts Default, Administrator, Developer, Replicator, and the predefined groups Everybody, Administrators, Developers, Replicators, and Anonymous. You cannot delete these user accounts and groups; therefore, make sure the *internal* accounts and groups have the correct definitions.



An exception to the above diagram is that all internally-defined users are members of the internally-defined Everybody group.

About User Groups and Package Replication

Although Integration Server is distributed with a predefined Replicator account, you can use a different account for package replication. As long as the subscription requester specifies an account that is a member of a group that is assigned to the Replicators ACL, that user can perform replication.

When publishing a package to another server, the publishing server uses the account specified by the subscription requester. For example, if the subscription requester (either the publisher or the subscriber) specified account DEPT01, the publisher will log into the subscriber server as DEPT01. DEPT01 must be a member of a group that is assigned to the Replicators ACL on the subscriber server.

Refer to [“Copying Packages from One Server to Another” on page 673](#) for more information about package replication.

About Granting Administrator Privileges to External Users

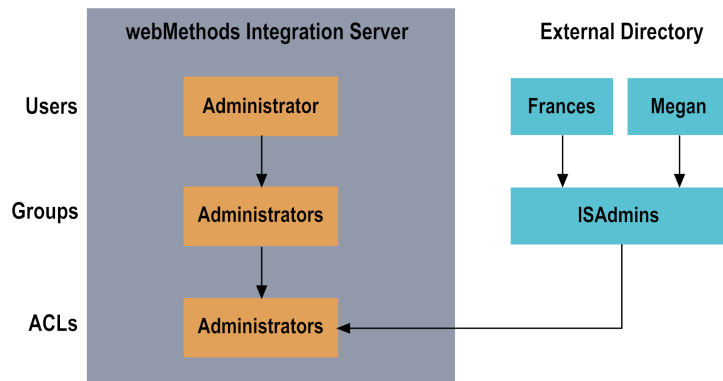
The Administrators ACL controls who has administrator privileges. Because you cannot assign externally defined users to internally-defined groups, you cannot grant externally defined users administrator privileges by assigning them to the internally-defined Administrators group. Instead, you need to set up an externally defined group for administrators. Then, add the externally defined group of administrators to the Administrators ACL.

To make a group of central users IS Administrators, you will need to add their group or role to the following ACLs:

- Administrators ACL
- Default ACL
- Developers ACL
- Internal ACL
- Replicators ACL
- Anonymous ACL (if their role/group is not part of this already)

Note:

If you configured Integration Server to use central user management, the Anonymous ACL automatically includes the My webMethods users role.



Granting Administrator Privileges to an Externally Defined User

➤ To grant administrator privileges to an externally defined user

1. Set up an externally defined user account for the user if one does not already exist.
2. Set up an externally defined administrators group if one does not already exist.

Important:

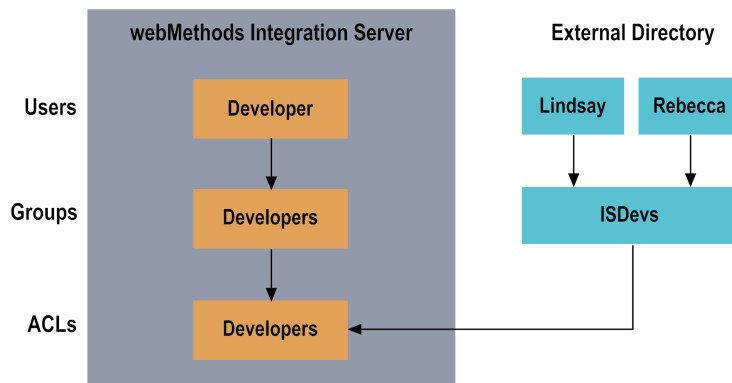
Do not name the externally defined group "Administrators". The name of the group must not be the same name as any internally-defined group.

3. Make the externally defined user a member of the externally defined administrators group (ISAdmins in the picture above).
4. Update the Administrators ACL to include the externally defined administrators group in the **Allowed** list.

Refer to [“Allowing or Denying Group Access to ACLs” on page 506](#) for information on how to include externally defined administrators to the Allowed list.

Granting Developer Privileges to External Users

The Developers ACL controls who can connect to the Integration Server from Software AG Designer to create, modify, and delete services that reside on the server. Because you cannot assign externally defined users to internally-defined groups, you cannot grant externally defined users developer privileges by assigning them to the internally-defined Developers group. Instead, you need to set up an externally defined group for Designer. Then, add the externally defined group to the Developers ACL.



➤ To grant developer privileges to an externally defined user

1. Set up an externally defined user account for the user if one does not already exist.
2. Set up an externally defined developers group if one does not already exist.

Important:

Do not name the externally defined group "Developers". The name of the group must not be the same name as any internally-defined group.

3. Make the externally defined user a member of the externally defined developers group (ISDevs in the picture above).

- Update the Developers ACL to include the externally defined developers group in the **Allowed** list.

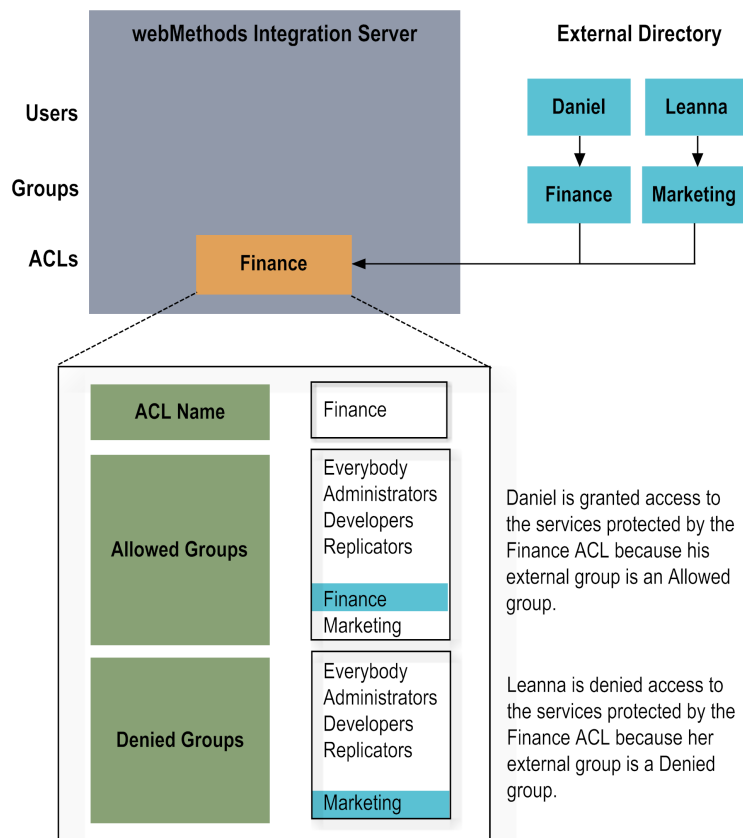
Refer to [“Allowing or Denying Group Access to ACLs” on page 506](#) for information on how to include externally defined developers to the Allowed list.

Granting Access to Services and Files to External Users

You create ACLs that control access to services and files and assign them to the specific services and files that you want to protect.

To grant access to a service or file, the server first uses internally-defined information to determine whether the client is a member of allowed or denied groups listed in the ACL. If the server cannot find the information internally, it obtains externally defined information to determine if the ACL allows or denies access.

If you want to allow an externally defined user access to a service or file, update the ACL that protects the service or file to identify the external user's group or role as an Allowed group in the ACL. Similarly, if you want to explicitly deny an externally defined user access to a service or file, update the ACL that protects the service or file to identify the external user's group or role as a Denied group in the ACL.



35 Managing Packages

■ Using Packages	654
■ How the Server Stores Package Information	657
■ Finding Information about Your Packages	660
■ Working with Packages	667
■ Copying Packages from One Server to Another	673
■ Using a Package Class Loader	697
■ Hot Deployment of Packages	697
■ Automatic Package Deployment	701

Using Packages

A package contains a set of services and related files, such as specifications, document types, and DSPs. When you add a service, specification, document type, or DSP to the webMethods Integration Server, you must add it to a package. Use a package to group services and related files.

By placing related files in a package, you can easily manage all the services and files in the package as a unit. For example, you can make them all available, disable them, refresh them, or delete them with one action. Additionally, if you have more than one Integration Server installed, you can use package management features to copy some or all services and files in a package to another server.

You can group your services using any package structure you choose, though most organizations group services into packages by function or application. For example, you might put all purchasing-related services in a package called “PurchaseOrderMgt” and all time-reporting services into “TimeCards.”

Important:

Every service on the server must belong to a package. Before you can make a service available for execution, you must load the package to which it belongs.

Access to a package and its contents is controlled through Access Control Lists (ACLs). Using ACLs, you control who can display a package from the Integration Server Administrator and Designer, who can edit the contents of a package, and who can execute services contained in the package. For more information about protecting packages, see [“Controlling Access to Resources with ACLs” on page 501](#).

You can associate a package with a specific port so that when you replicate the package, it continues to use a port with the same number on the new server. See [“About Ports” on page 151](#) for more information about associating a package with a port.

Predefined Packages

The following packages are predefined and installed by default on each Integration Server instance. They are also included in the package repository.

Package	Description
Default	<p>Integration Server looks in this package if a user accesses the server without specifying a package name (for example, by using the URL <code>http://localhost:5555</code>). You can also use it to store elements you create without first creating a package.</p> <p>Note: Integration Server searches the <code>pub</code> directory in the Default package for an <code>index.dsp</code> or <code>index.html</code> file. As shipped, the <code>pub</code> directory contains an <code>index.html</code> file that points the user to an <code>index.dsp</code> file in the <code>WmRoot</code> package. This <code>index.dsp</code> file loads the Integration Server Administrator. To prevent a user from inadvertently accessing the Integration Server Administrator, you can edit the <code>index.html</code> file in <code>Default/pub</code> and change it to point to an innocuous page. To prevent</p>

Package	Description
	a user from seeing or accessing all DSP files on Integration Server, you can use the <code>watt.server.displayDirectories</code> server configuration parameter. See “Stage 7: Setting Up Security” on page 1013 for more information.
WmPublic	This package contains services that you can call from your client applications and services. See the <i>webMethods Integration Server Built-In Services Reference</i> for more information.
WmRoot	This package provides core Integration Server functionality and auxiliary files. Important: Do not alter or delete this package.
WmAdmin	This package provides the functionality and files for the Integration Server Administrator API. For more information about the Administrator API, see “Integration Server Administrator API” on page 971 API. Important: Do not alter or delete this package.
WmART	This package supports webMethods 6 or later adapters. Important: Although the WmArt package is installed by default, if you are not fully licensed to use this package, it will be hidden from view, and you will not be able to build or run custom adapters, or use any transaction services.
WmXSLT	This package contains services that you use to transform XML data from one format or structure to another. See the <i>webMethods Integration Server Built-In Services Reference</i> and <i>webMethods Service Development Help</i> for more information.
WmJSONAPI	This package contains services that support REST V2 resources to become JSON API compliant.

The packages in the table below provide services that enable you or other webMethods products to perform certain tasks.

This package	Contains services that	Refer to
WmARTEExtDC WmISEExtDC WmTNEExtDC	Infrastructure Data Collector uses to discover and monitor adapters installed on Integration Server, Integration Server itself, and Trading Networks Server, respectively.	<i>Administering webMethods Optimize</i>

This package	Contains services that	Refer to
WmAssetPublisher	Integration Server uses to extract and publish metadata about its services to CentraSite Metadata.	<i>webMethods BPM and CAF Workspace Metadata Help</i>
WmCloud	Integration Server uses to share services with webMethods Integration Cloud.	<i>Configuring On-Premise Integration Servers for webMethods Cloud</i>
WmDesigner	Support business processes modeled in Software AG Designer.	<i>Software AG Designer Online Help</i>
WmFlatFile	Flow services or the File Polling processing service can call to initially accept and consume inbound flat files.	<i>webMethods Integration Server Built-In Services Reference</i>
WmMobileSupport	Contains services that support data synchronization solutions between mobile devices and backend applications.	<i>Developing Data Synchronization Solutions with webMethods Mobile Support</i>
WmOptimize	Support business processes monitored using webMethods Optimize.	Optimize documentation
WmPRT	Support business processes executed using the webMethods Process Engine.	<i>Administering webMethods Process Engine</i>
WmTaskClient	Support tasks developed using the webMethods Task Engine.	<i>webMethods Task Engine User's Guide</i>
WmWin32	You can use for Integrated Windows authentication on Integration Server. Note: This package is deprecated in Integration Server 7.1	"Using NTLM Authentication when Integration Server Acts as the Client" on page 1191

If you want to streamline Integration Server (for example, because you are using it only to host adapters), you can disable many of the packages described in above.

Important:

Do not disable, delete, or alter the following packages: Default, WmPublic, WmRoot.

You can disable the following packages:

Package	Restriction
WmAdmin	Important: Do not disable the WmAdmin package if you use the Integration Server Administrator API.
WmART	Important: Do not disable this package unless you are also going to disable WmARTEExtDC, or WmARTEExtDC will not load.
WmAssetPublisher	None.
WmARTEExtDC WmISEExtDC WmTNEExtDC	Important: Do not disable the WmISEExtDC package unless you are also going to disable WmTNEExtDC and WmARTEExtDC, or those packages will not load.
WmFlatFile	None.
WmPRT WmDesigner WmTaskClient	None.
WmMobileSupport	Before disabling the WmMobileSupport package, suspend all mobile sync components that are enabled.
WmOptimize	None.
WmXSLT	None.
WmJSONAPI	None.

For instructions on disabling a package, see [“Disabling a Package” on page 670](#).

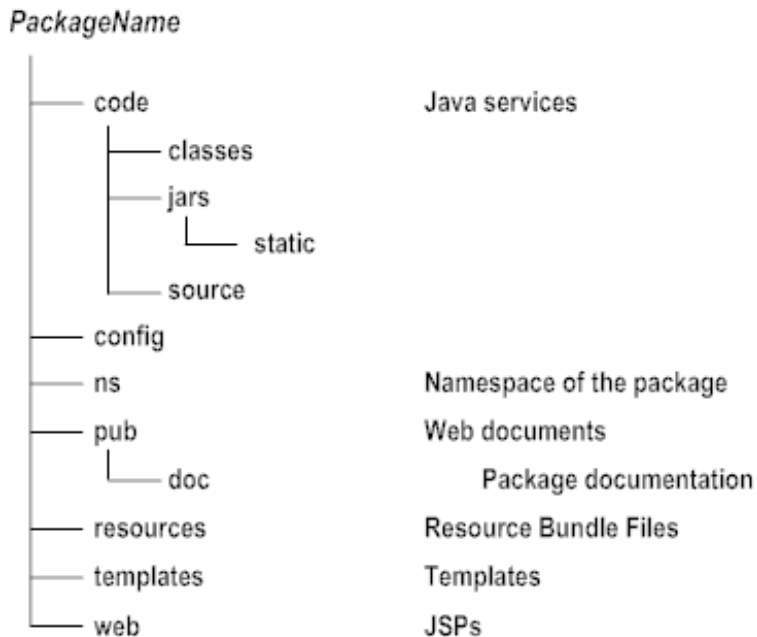
The Package Repository

The *Software AG_directory* \IntegrationServer\packages directory serves as a repository for packages that you can install on any Integration Server instance. The repository includes predefined packages and many other packages that you can use to perform certain tasks. For information about how to install or update packages from the package repository, see [“Updating Packages on a Server Instance” on page 70](#).

How the Server Stores Package Information

The server physically stores package information in the *Integration Server_directory* \instances*instance_name*\packages directory. The server creates a new subdirectory for each package. The name of the subdirectory is the name of the package. For example, if a package is named “TimeCards,” the server creates the *Integration Server_directory* \instances*instance_name*\packages\TimeCards directory to hold the files for the package.

When you create a new package, the server creates the following subdirectories to hold all the files associated with the package:



- **The code subdirectory** holds the Java and C/C++ services that belong to this package. Within the code subdirectory are the classes, jars, static, source, and lib subdirectories:
 - **The classes subdirectory** is for Java classes for the Java and C/C++ services.
 - **The jars subdirectory** is for Java classes that are packaged together in jar files.
 - **The static subdirectory** is also for Java classes that are packaged together in jar files. Place the jar files in this location when you want to make them available to other packages in the Integration Server and also to packages in other Integration Server systems.

When you place jar files in the static subdirectory, then at startup the Integration Server automatically loads these files to the server classpath. Also, the jar files are available to other packages even when the immediate package is disabled.

Note:

The Integration Server does not automatically create the static subdirectory when you create a new package. This subdirectory is user defined.

- **The source subdirectory** is for the source of Java services.
- **The libs subdirectory** (not shown here) holds DLLs or specialized libraries that the Java and C/C++ services use.

Note:

The Integration Server does not automatically create the libs directory because the directory's existence prevents you from reloading a package without restarting the

server. You cannot reload a package that uses shared libraries; you must restart the server.

For ease of administration, place services that use shared libraries in the same package.

- **The config subdirectory** might hold following files:
 - **listeners.cnf.** This file contains information about what ports you have defined for the package and what services you can access through each port. For more information about configuring ports and allowing access to services through a port, see [“Configuring Ports” on page 149.](#)
 - **urlalias.cnf.** This file contains definitions for HTTP URL aliases associated with services in the package. If there are no aliases, the file does not exist. Do not manually update a urlalias.cnf file. For more information about HTTP URL aliases, see [“Setting Up HTTP URL Aliases” on page 423.](#)
- **The ns subdirectory** holds flow services, specifications, document types, schemas, triggers, adapter notifications, adapter documents, adapter services, adapter connectors, and code fragments for Java services.
- **The pub subdirectory** holds web documents for the package. For instructions on how to access the web documents for a package, see [“Displaying Documentation for a Package” on page 667.](#)
- **The doc subdirectory** holds documentation for the package.
- **The resources subdirectory** holds resource bundles *<bundle.properties>*, such as application data (not user data), which is kept separate from the Integration Server application. The following items represent typical resources inside a bundle:
 - Icons
 - Window positions
 - Dialog box definitions
 - Program text
 - Menus

You can easily modify and update various aspects of the Integration Server without reinstalling the entire application. A Japanese language pack for the Integration Server is an example of a resource bundle that contains language and image files for the Japanese version of the server.

- **The templates subdirectory** holds output templates that are associated with this package.
- **The web subdirectory** holds JSPs that are associated with this package.

Manifest File

Each package has a manifest file. It contains:

- **Indication of whether the package is enabled or disabled.** The server does not load disabled packages at server initialization and you cannot access elements that reside in disabled packages.
- **List of startup, shutdown, and replication services, if any, for the package.** For more information about startup, shutdown, and replication services and how to identify them, see [“Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710](#).
- **Package description.** A brief description of the package.
- **Version information.** Package version and build number. Also included is the JVM version under which the package was published.
- **Patches applied.** A list of patches that have been applied to the package. These are names or numbers that are meaningful to your installation, possibly obtained from your problem tracking system.
- **Package dependencies, if any, for the package.** For a specific package, the developer can identify other packages that the server should load before it loads the elements in a particular package. In other words, the developer can identify when one package depends on another. For information, see [“Displaying Information about a Package” on page 664](#) and *webMethods Service Development Help*.
- **Target package name.** Name of the package.
- **Publishing server.** The Integration Server that published the package. If the package has not been published, this field contains None.

The manifest for a package is in the manifest.v3 file in the top directory for the package.

Finding Information about Your Packages

The server displays a variety of information about your packages. The section describes the information that is available and the procedures to use to display the information.

Information	Refer to page:
List of all of the packages that reside on your server	“Viewing the Packages that Reside on Your Server” on page 661
List of specified packages that reside on your server	“Viewing the Packages that Reside on Your Server” on page 661
Status of whether the server successfully loaded the package or not	“Determining Whether the Server Successfully Loaded the Package” on page 663
Status of whether the package is enabled or disabled	“Determining Whether the Package Is Enabled or Disabled” on page 664
Version number of the package	“Displaying Information about a Package” on page 664

Information	Refer to page:
Number of elements in a package that the server successfully loaded into memory	“Displaying Information about a Package” on page 664
Name of Access Control List (ACL) that controls which users can list the package	“Displaying Information about a Package” on page 664
List of elements in a package that the server failed to load into memory	“Displaying Information about a Package” on page 664
List of load errors for the package	“Displaying Information about a Package” on page 664
List of startup, shutdown, and replication services in a package	“Displaying Information about a Package” on page 664
List of packages on which a package depends	“Displaying Information about a Package” on page 664
List of servers that subscribe to this package	“Displaying Information about a Package” on page 664
Documentation for the package	“Displaying Documentation for a Package” on page 667

To display a list of HTTP URL aliases associated with a package, use the **Settings > HTTP URL Aliases** screen. For more information about HTTP URL aliases, see [“Setting Up HTTP URL Aliases” on page 423](#).

Viewing the Packages that Reside on Your Server

The main package management screen of the Integration Server Administrator lists all packages that reside on your server. It also displays whether the server successfully loaded the package and whether the package is enabled.

Note:

The server displays only packages to which you have List access.

➤ To view the packages that reside on the server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.

Filtering the List of Packages

By default, the main package management screen lists all the packages that reside on your server. You can use the filter to limit the packages to be displayed, making the list shorter and more manageable. You can manage the **Packages List** in the following ways:

- Filter the default list of packages by specifying a full or partial package name, and then including or excluding packages that match the specified criteria.
- Narrow down a list by filtering the results again.

Tip:

Click **Show All Packages** to disable filtering and restore the default list of all packages on the server.

➤ **To filter the package list**

1. In the **Packages** menu of the Navigation panel, click **Management**. The **Packages List** will display all the packages on your server.
2. Click **Filter Packages**. The filtering options will appear above the **Packages List**.

Note:

When **Filter Packages** is enabled, any changes to the Integration Server (such as new packages, etc.) will not be reflected in the **Packages List**. When you click **Show All Packages** and return to normal mode, the list will be updated.

3. Select some or all of the following options:

Option	Description
Filter criteria	The string you want to submit to the filter. By default, packages with names that match the string are included in the results. Filter criteria can be literals or a combination of literal and wild-card characters. The "*" (asterisk) and "?" (question mark) are the only supported wild-card characters. Leaving the filter criteria blank includes all packages.
	<p>Important:</p> <p>The package names in the Filter criteria field are case-sensitive. For example, if you enter "wma*", the filter will ignore any packages beginning with "WmA".</p>
Include Enabled	Specify whether to include only packages that are enabled (those with Yes in the Enabled column of the Packages List), only those that are disabled (No is in the Enabled column), or to include both enabled and disabled packages.
Include Disabled	
Include Both	
Filter on result	Enable this option when you have already filtered the list and you want to re-filter the results, rather than the default list.

Option	Description
Exclude from result	Enable this option to display the packages that do <i>not</i> match the Filter criteria , rather than the packages that do match.

- Click **Submit**. Only the packages which match the filter options will be displayed.

Refining the Filtered Package List

➤ To filter packages from an already filtered Package List

- Filter the **Packages List** as described in “[Filtering the List of Packages](#)” on page 661. The packages which match the filter will be displayed.
- Enable the **Filter on result** mode. This limits the search to just the currently displayed list of packages, rather than the default list of all the packages on the server.

Note:

You can also enable the **Exclude from result** option to display the packages that do *not* match the Filter criteria, rather than the packages that do match.

- Enter the new filter criteria and click **Submit**. Repeat as many times as necessary, being sure to enable the **Filter on result** mode each time.

Determining Whether the Server Successfully Loaded the Package

The server displays a status in the **Loaded?** column of the **Packages** screen.

Status	Indicates that
✓ Yes	The server successfully loaded all elements associated with the package. The elements in the package are available. The server also displays this status if the package is empty.
✓ Partial	The server did not load one or more of the elements associated with the package. The elements that the server successfully loaded are available. For instructions on how to determine which elements the server did not successfully load and why, see “ Displaying Information about a Package ” on page 664.
No	The server did not load any of the elements associated with the package. None of the elements are available. For instructions on how to determine why the server could not load the elements, see “ Displaying Information about a Package ” on page 664.

When the server is started, it automatically loads into memory all elements that are in enabled packages. If a package is disabled at startup, the server loads the elements into memory when the

package is enabled. You can manually reload a package if necessary. For instructions on reloading a package, see [“Reloading a Package” on page 669](#).

Determining Whether the Package Is Enabled or Disabled

The server displays a status in the **Enabled** column of the **Packages** screen. The status indicates whether the package is enabled or disabled. A package must be enabled before the server allows clients access to the services in the package.

Status	Indicates that
✓ Yes	The package is enabled and clients can access the elements in the package.
No	The package is disabled and clients cannot access the elements in the package.
Warnings	Some of the elements in the package encountered warnings, but were loaded and are available for use. To learn which elements caused warnings, look at the Load Warnings list at the bottom of the screen.

For instructions on enabling and disabling packages, see [“Enabling a Package” on page 670](#) and [“Disabling a Package” on page 670](#).

Displaying Information about a Package

You can display information about any package installed on Integration Server.

➤ To display information about a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. In the **Package List** area of the screen, click on the name of the package for which you want to display information.
4. The server displays the **Packages > Management > *PackageName*** screen. For a detailed description of these fields, see [“Package Information” on page 664](#).

Package Information

Integration Server maintains the following information about each installed package.

Field	Description
Package Name	Name of the package.
Version	Version number of the package.
Build	A number that a developer assigns to a package each time it is regenerated. For example, a developer might generate version 1.0 of the Ordering package 10 times and assign build numbers 1,2,3,...10. These build numbers are generally used to identify the generations of a package in a development environment.
Minimum Version of JVM	<p>Minimum version of the Java Virtual Machine (JVM) required to run this package.</p> <p>The Integration Server on which the package is installed must run in a JVM with a major version that is the same or higher than the JVM version required by the package. For example, if the specified JVM version for a package release is 1.8.0_32, the installing Integration Server must be running in a JVM that is 1.8 or higher. If the major version of the Integration Server JVM is less than the JVM version required by the package, Integration Server installs but does not activate the package.</p>
Package List ACL	The Access Control List assigned to the package. Users associated with this ACL can see the package listed on the Integration Server Administrator or Designer. To see the folders and elements contained in the package, a user must have List access to the folders and elements themselves.
Patches Included	A list of patches that have been applied to this release of the package. These are numbers that are meaningful to your installation, possibly obtained from your problem tracking system.
Description	A description of the package and its intended use.
Publisher	<p>The name of the company, organization, or server that published the package.</p> <p>Note: By default, the Integration Server automatically enters the publishing server name in this field only when you create a package release.</p>
Created on	<p>Date, time, and year in which the package was created.</p> <p>Note: By default, the Integration Server automatically enters the date, time, and year in this field only when you create a package release.</p>
Elements Loaded	Number of elements that the server successfully loaded. To view the elements that the server has successfully loaded, click the Browse services in < PackageName > link.

Field	Description
Elements Not Loaded	Number of elements that the server failed to load. If the server failed to load one or more elements, the Load Errors section of the screen lists the elements that it could not load, along with the reason.
Startup Services	List of the services that you or another administrator have identified as startup services. For more information about startup services, refer to “Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710.
Shutdown Services	List of the services that you or another administrator have identified as shutdown services. For more information about shutdown services, see “Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710.
Replication Services	List of the services that you or another administrator have identified as replication services. For more information about replication services, see “Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710.
Packages on which this package depends	List of the packages the server must load before it loads this package. For more information about package dependencies, see <i>webMethods Service Development Help</i> .
Packages that depend on this package	List of packages that depend on this package. If you disable the package, these packages will be affected.
Subscribers	List of other Integration Servers that subscribe to this package. For information on how to copy packages from one server to another, how to subscribe to packages, and how to publish packages to another server, see “Copying Packages from One Server to Another” on page 673.
Load Errors	Displays a list of elements that generated errors and could not be loaded onto the server when the package was installed. When some elements do not load, the load status for the package becomes Partial .
Load Warnings	Displays a list of elements that generated warnings when the package was installed. The server was able to load the packages, despite the warnings. When package elements are loaded with warnings, the load status for the package becomes Warnings .
Patch History	A list of patches or partial packages that have been applied to this release of the package.

Displaying Information about Services and Folders in a Package


You can browse a list of services or folders in a package. See [“Finding Information about Services and Folders”](#) on page 705.

Displaying Documentation for a Package

You can document the function of a package and its elements in web documents that the Integration Server will serve. Place the web documents in the `pub` subdirectory for a package.

Be sure to create an `index.html` file that holds the home page for the package and contains links to the other web documents for the package.

➤ To access the home page for a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel click **Management**.
3. Click the home  icon for the package.

Accessing a Web Document for a Package

➤ To access any web document for a package

1. Make sure the package is enabled. (See [“Determining Whether the Package Is Enabled or Disabled”](#) on page 664 for instructions.)
2. Enter the URL for the web document. The URLs for the web documents have the following format:

`http://host:port/PackageName/Docname`

where:

<i>host:port</i>	is the server name and port address of the Integration Server
<i>PackageName</i>	is the name of the package in which the web document resides. If you do not specify a package name, the server looks in the <code>Pub</code> directory of the Default package.
<i>DocName</i>	is the name of the web document. If you do not specify a document name, the server displays the <code>index.dsp</code> or <code>index.html</code> file in the <code>Pub</code> directory of the specified package.

Working with Packages

You can perform the following tasks that act on all the files in a package as a unit:

Use this function	When you want to	Refer to
Create	Create a new package. Developers create packages using Designer. See <i>webMethods Service Development Help</i> for more information.	“Creating a Package” on page 668
Activate	Use a package that you manually moved into the Server/packages directory without having to restart the server.	“Activating a Package” on page 668
Reload	Reload the services in the package into memory without having to restart the server.	“Reloading a Package” on page 669
Enable	Enable a package that you previously disabled.	“Enabling a Package” on page 670
Disable	Disable access to a package without deleting it.	“Disabling a Package” on page 670
Delete	Delete all services and related files in a package.	“Deleting a Package” on page 671
Recover	Recover the services and related files from a package that you previously deleted.	“Recovering a Package” on page 672
Archive	Make a working copy of a package without making it generally available to others through a release. You might use this copy as a backup.	“Archiving a Package” on page 672
Copy	Copy a package from one server to another.	“Archiving a Package” on page 672

Note:

You can also manage packages by using a set of built-in services. See the *webMethods Integration Server Built-In Services Reference* for more information.

Creating a Package

When a developer wants to create a new grouping for services and related files, the developer creates a package. This creates an empty container into which your developers can store services and related files. When a developer creates a package, the server builds the directory structure of the package as described in [“How the Server Stores Package Information” on page 657](#). For instructions on creating a package, see *webMethods Service Development Help*.

Activating a Package

There may be times when a package is installed on your Integration Server but is not active. When a package is active, it is “officially recognized” by Integration Server and displayed in the **Package**

List on the **Package Management** screen. When a package is inactive, it exists in the Packages directory, but is not officially recognized by the server.

Possible reasons for a package being inactive are:

- You manually installed the package while Integration Server was running.
- Another server published the package to your server, but the package requires a version of the JVM that is higher than the version on your server. A subscribing server will not activate a package under these circumstances.

The subscribing Integration Server must run in a JVM whose major version is the same or higher than the JVM version required by the package. For example, if the specified JVM version for a package release is 1.8.0_32, the installing Integration Server must be running in a JVM that is 1.8 or higher.

- The package you installed has dependencies on another package that either does not exist on your server or is disabled. If the package is disabled, the server installed the package but did not activate it. You can activate the package when the dependencies are satisfied.

The package will not be available until either you restart the server or you activate the package.


➤ To activate a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click **Activate Inactive Packages**.
4. In the **Inactive Packages** area, select the package you want to activate from the pull-down menu and click **Activate Package**.

Reloading a Package

If the server is running when a developer changes a Java service or flow service, you must reload the package in which the service is contained for the changes to take effect. Reloading the package invokes the VM class loader to reload the package's Java services and reloads the flow services into memory. Developers can also reload a package from Designer.

➤ To reload a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click the reload icon  in the **Reload** column for the package.


4. The green check mark icon in the **Loaded?** column indicates whether the server loaded the package successfully. For more information, see [“Determining Whether the Server Successfully Loaded the Package”](#) on page 663.

Enabling a Package

To allow clients access to the elements in a package, you must ensure the package is enabled. Before the server can access an element in a package, the package must be enabled and the element must be loaded. By default, packages are enabled.

When you enable a disabled package, the server loads the elements in the package into memory.

> To enable a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click **No** in the **Enabled** column for the package you want to enable. The server issues a prompt to verify that you want to enable the package. Click **OK** to enable the package. When the package is enabled, the server displays a  icon and **Yes** in the **Enabled** column.

Disabling a Package

When you want to temporarily prohibit access to the elements in a package, disable the package. When you disable a package, Integration Server unloads all of its elements from memory. Be aware that if you disable a package while one or more services in the package are executing, those services will most likely fail. Integration Server does not wait for in-progress services to finish before disabling a package.

Important:

Never disable the WmRoot package. Integration Server uses the services in this package.

> To disable a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel click the **Management** link.
3. Click the green check mark on in the **Enabled** column for the package you want to disable. The server issues a prompt to verify that you want to disable the package. Click **OK** to disable the package. When the package is disabled, the server displays **No** in the **Enabled** column.

Note:

The server retains the access status of a package (enabled or disabled) across server restarts. When you start the server, the server does not load elements in disabled packages.

Deleting a Package

When you no longer need the services and files in a package, you can delete the package. When you delete a package, all the elements of the package (services, specifications, document types) become unavailable.



When you delete a package, you can optionally select to save a copy of the package. If you save a copy, the server copies the package to the *Integration Server_directory* \instances*instance_name*\replicate\salvage directory before deleting the package from the *Integration Server_directory* \instances*instance_name*\packages directory. If needed, you can recover the package at a later time. For instructions on recovering a deleted package, see [“Recovering a Package” on page 672](#).

Important:

Never delete the WmRoot package. The Integration Server uses the services in this package.

When you delete a package from the command prompt, you have the option of deleting a single package or a list of packages. For more information about deleting a package from the command prompt, see [“Deleting Packages from a Server Instance” on page 72](#).

> To delete a package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel click **Management**.
3. If you want to save a copy of the package so you can recover it later if necessary, check the  icon in the row that corresponds to the package you want to delete.
4. If you do *not* want to save a copy, click the  icon in the row that corresponds to the package you want to delete.
5. Click **Delete**. The server asks you to confirm you want to delete the package. Click **OK**.

Note:

When you delete a package using the delete or safe delete options, Integration Server does not delete the jar files in the code/jars and code/jars/static directory. The jar files in code/jars folder are loaded by the Package class loader and jars files in code/jars/static folder are loaded by the Integration Server class loader. These jar files are locked by Integration Server and can only be deleted when Integration Server is shut down. If you install a new version of a package without removing the jar files for the old version of the package, Integration Server uses the old jar files with the new package version. This can result in inconsistent or unexpected behavior. To avoid this problem, before installing a new version of a deleted package, shut down Integration Server and delete any jar files left in the *Integration Server_directory* / instances/*instanceName*/

packages/*packageName*/code/jars and *Integration Server_directory* / instances/*instanceName*/packages/*packageName*/code/jars/static directories.

Recovering a Package

If you deleted a package from a server instance using the **Safe delete** option and you need the package again, you can recover the package.

Note:

If you did not use the **Safe delete** option, it is still possible to restore the package from the webMethods Integration Server package repository. If the package exists in the repository, you can perform a package update from the command prompt. For instructions on updating packages from the command prompt, see [“Updating Packages on a Server Instance” on page 70](#).

➤ To recover a package after using Safe Delete

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel click **Management**.
3. Click **Recover Packages**.
4. In the **Recover Packages** area, select the package you want to recover from the pull down.
5. If you want the Integration Server to automatically activate the package when it is recovered, select the **Activate Upon Recovery** check box.
6. Click **Recover**.

Archiving a Package


There may be times when you want to make a copy of a package without making it generally available. For example, you might want to back it up or send it to someone with whom you do not have a publisher/subscriber relationship.

Note:

An alternative way of creating a backup is using the `pub.packages:backupPackage` service. However, when you use the `backupPackage` service, the package metadata of the backed up package is the same as the original package. For example, the creation timestamp will reflect the creation timestamp of the original package. For more information, see the *webMethods Integration Server Built-In Services Reference*.

➤ To archive a package

1. Open the Integration Server Administrator if it is not already open.

2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Locate the package you want to archive in the **Package List**, and click the  icon.

The server displays a screen from which you specify the files you want to archive, the type of archive (full or patch), and version information. See [“Specifying File and Version Information for a Release or Archive” on page 684](#) for instructions on specifying this information.

Note:

You can also archive a package automatically after installing the package. For more information, see [“About Installing Packages Published by Another Server” on page 695](#).

Copying Packages from One Server to Another

Use package replication to copy (publish) packages from one Integration Server to another. If you have a clustered environment, this feature is useful to quickly replicate new and updated packages across all servers in the cluster. It is also a convenient way to distribute a package from one server to another anywhere on the web.

Note:

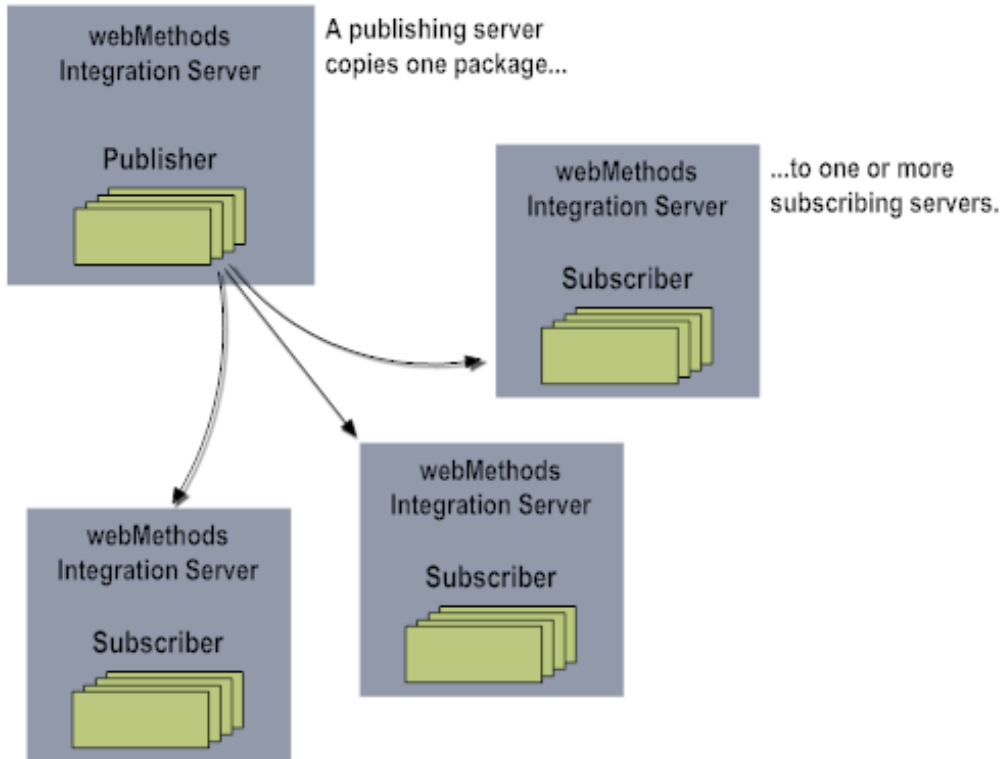
Using Software AG Designer, you can copy a package and its contents to another Integration Server by performing a copy or a drag-and-drop action. Copying packages using Designer provides a quick way to test a set of services and their supporting files, for example, in a remote environment. This method is useful in single development environments where change control is not crucial. In a production environment, however, using the package replication function is recommended.

Note:

If you want to make a copy of package, for example to make a backup, without sending it to another server, see [“Archiving a Package” on page 672](#).

Overview of Package Replication

During replication, a single Integration Server sends (publishes) a specified package to one or more recipient servers. The server on which the package originates is referred to as the *publisher*, and the recipients are referred to as *subscribers*.



Subscribing servers receive the package in their inbound directory (*Integration Server_directory \instances\instance_name\replicate\inbound*). To activate the new package, an administrator on the subscribing server must install the package after it arrives. (This procedure is explained in [“About Installing Packages Published by Another Server”](#) on page 695.)

Either a publisher or a subscriber can request a subscription. A publisher can send (push) the package and the subscriber can request (pull) the package.

Before you send a package to another server, you must create a *release*. When you create a release, the server creates a distribution file that contains the package and information about the package, and makes the package available to subscribers.

You can have multiple releases for a given package. For example, you might have separate releases for versions 1.0, 1.1, and 1.2 of a given package. Or, you might use different releases to separate packages for different audiences. Each release must have a unique name.

Important:

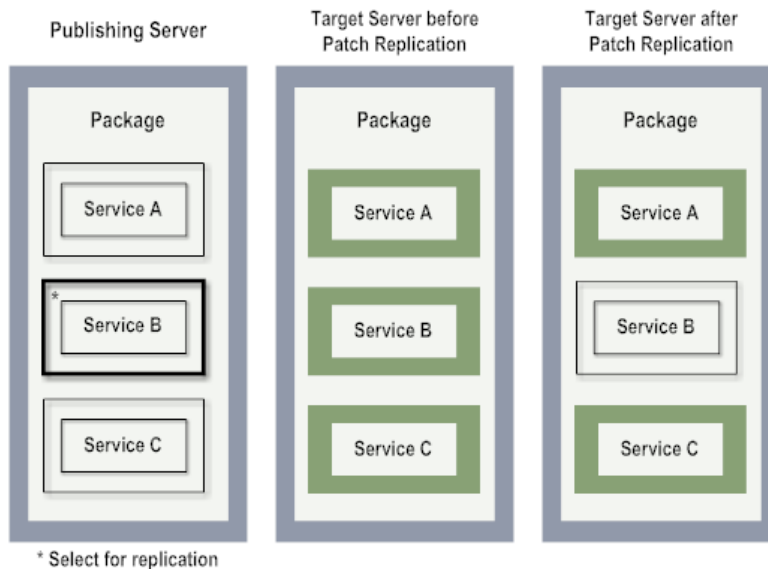
If you have multiple releases of a given package and one or more subscribers have specified the automatic pull feature, those subscribers will receive *all releases* of a package when a new release of it becomes available. For more information about the automatic pull feature, see [“The Subscribing Server”](#) on page 687.

A release can contain the complete package (a *full* release) or just patches to the package (a *patch* release). Typically you will publish a full release when you have made major changes to the package and use patches just to correct problems with a package.

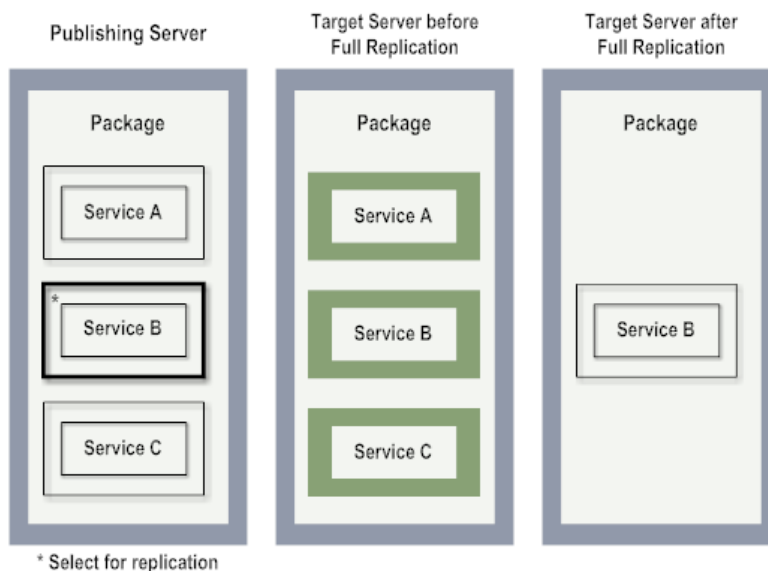
With a full release, the new package entirely replaces the old package on the subscriber's server. With a patch release, the files in the patch release replace the versions of those files in the target package; all other files in the target package remain intact.

In addition to specifying a full or patch release, you can select all files to go in the release or just some.

The following diagram illustrates how a patch release replaces files:



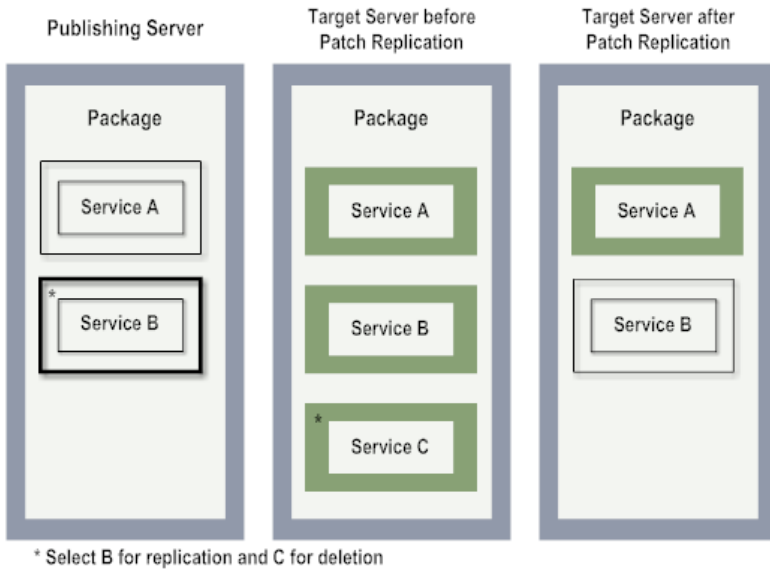
The following diagram illustrates the results if you selected a single service for replication and specified a *full* release instead.



Most often you will select all files and specify a full release, or select some files and specify a patch release. There might be times however when you want to select just some files and specify a full release. For example, there might be files in a package, such as internal documentation files, that

a developer does not want released to others. Selecting all files except the extraneous ones and specifying a full release results in a package that contains just the desired files.

There might be other times when you want to replace some files, leave others intact, and delete others. To achieve this greater level of control, you can perform a patch replication and specify files to copy and files to delete. Files that you do not specify for copying or deletion remain intact. In the following example, we want to leave Service A intact, replace Service B, and delete Service C from the target package.



The following shows what you must specify on the **Specify Files for the Release** screen to accomplish this task:

Specify Files for the Release

Files available in SimplePackage:

- ns/MyFolder/ServiceA/flow.xml
- ns/MyFolder/ServiceA/flow.xml.bak
- ns/MyFolder/ServiceA/node.ndf
- ns/MyFolder/ServiceB/
- ns/MyFolder/ServiceB/flow.xml
- ns/MyFolder/ServiceB/flow.xml.bak
- ns/MyFolder/ServiceB/node.ndf
- ns/MyFolder/ServiceC/
- ns/MyFolder/ServiceC/flow.xml
- ns/MyFolder/ServiceC/flow.xml.bak
- ns/MyFolder/ServiceC/node.ndf
- ns/MyFolder/node.idf

Files to Include

- All files
- Selected files
- All except selected files
- Files specified by filter:
- All except files specified by filter: (ex: *.java, *.class)

Files to Delete from Target Package (For distributing upgrades only)

One file name per line. Use ";" to separate multiple file names.

```
ns/MyFolder/ServiceC/flow.xml
ns/MyFolder/ServiceC/flow.xml.bak
ns/MyFolder/ServiceC/node.ndf
```

Select these files. They will replace the versions in the target package.

Click Selected files.

Type in these files. They will be deleted from the target package.

The Integration Server keeps track of package versions, Integration Server versions, and JVM versions so that during package installation the subscribing server can make sure the package being installed is compatible with the subscribing server's environment. The type of version checking performed depends on whether the release is a full or patch release.

Note:

If patch releases have been applied to a package, the developer can see the patch history when viewing the package from Designer. However, when the publisher publishes a full release of the package, the patch history is removed.

Version Checking

When the administrator on the subscribing server installs the package, the subscribing server performs some version checking:

Target server verifies that

Target JVM Version

The target server is running the same or a later version of the JVM, as specified during release creation. If this requirement is not met, the subscribing server issues a warning and installs the package but does not activate it.

The target Integration Server must run in a version of the JVM whose major version is greater than or equal to the JVM version required by the package. For example, if the specified JVM version for a package release is 1.8.0_32, the installing Integration Server must be running in a JVM that is 1.8 or higher. If the major version of the Integration Server JVM is less than the JVM version required by the package, Integration Server installs but does not activate the package.

See [“Activating a Package” on page 668](#) for instructions on activating a package.

Package Version	For a full release	For a patch release
	The version of the package on the target server is earlier than or the same as the package being installed. If this requirement is not met, package installation fails.	The version of the package on the target server exactly matches the version required by the release (as specified during release creation). If this requirement is not met, package installation fails.
	For example, if you create a new release and specify that it contains Version 2.0 of the wmExample package, the wmExample package on the target system must be release 2.0 or earlier.	For example, if you create a new release that contains a patch for wmExample package version 2.0, and you specify that the target package must be version 2.0, package installation will fail if the target package is not version 2.0.

Target server verifies that

This restriction prevents you from inadvertently installing an old version of a package over a newer one.

This restriction gives you greater control over how and where patches are applied. This is useful because patches are typically release dependent.

Who Can Subscribe?

Any Integration Server can subscribe to a package on another server if both servers allow it from a security perspective. Security for package replication is accomplished a number of ways:

- **Userid and password.** In order to send a package to a subscriber, the publisher must login to the subscriber by specifying a userid and password that exist on the subscriber.
- **ACLs.** The userid the publisher uses to log on to the subscriber must be a member of a group that is assigned to the Replicators ACL or higher on the subscriber.
- **SSL.** You can specify that the servers involved in package replication connect to each other using SSL.

The publisher maintains a list of subscribing servers for each package.

Subscriptions can be added by the publisher or the subscriber:

- **Publisher.** The administrator of a publishing server can use the publisher functions to add (or remove) subscribers to any package that *originates* on the publishing server (i.e., one to which you do *not* subscribe).
- **Subscriber.** The administrator of a remote Integration Server (the subscriber) can submit a subscription request to the publisher. When the publisher receives this request, it automatically adds that server to the subscription list for the requested package as long as authentication was successful. Subscribers can also issue cancellation requests (i.e., cancel their subscriptions) for packages to which they subscribe.

Guidelines for Using Package Replication

Keep the following guidelines in mind when using the package replication facility:

- Publishers and participating subscribers must use Integration Server Version 2.0 or later. For the Automatic Pull feature to work, they must be running Version 4.0 or later. If you are running version 4.0 or later of the Integration Server and publish to an earlier release of the Integration Server, the subscriber cannot perform a manual or automatic pull of a package. Instead, the subscriber must wait for the publisher to send the package.
- Any Integration Server can publish a package.
- Any Integration Server can subscribe to a package on another Integration Server.

- An Integration Server can be both a publisher of packages and a subscriber of other packages; however, it cannot be both a publisher and a subscriber of the *same* package.
- After setting up a subscription, if you delete the user account with which the subscription was set up (the account on the subscribing server that the publishing server uses to log on), the publisher will not be able to log into the subscribing server to send this package.
- Be careful when replicating a package that is associated with a port; the new port might decrease security on the target system. For example, suppose you replicate a package that is associated with an HTTP port at 5556. The replication process creates an HTTP port at 5556 on the target server. If the target server normally uses only HTTPS ports because of their greater security, then the new port presents a possible security hole on that server.

The Publishing Server

This section describes the tasks you perform when your server is participating in package replication as the publishing server:

Task	Refer to
Displaying the list of subscribers for a package	“Displaying Subscribers for a Specific Package” on page 679
Specifying subscribers for a package	“Adding Subscribers from a Publishing Server” on page 680
Updating subscriber information	“Updating Subscriber Information” on page 681
Removing subscribers for a package	“Removing Subscribers for a Package” on page 682
Publishing a package to subscribing servers	“Publishing a Package” on page 683
Specifying File and Version Information for a Release or Archive	“Specifying File and Version Information for a Release or Archive” on page 684

Displaying Subscribers for a Specific Package

Use this procedure to display the list of subscribers for a specific package on your server.

➤ To display the subscribers for a single package

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click the name of the package for which you want to view subscribers.

The server lists the subscribers to the package in the **Subscribers** field.

Displaying Subscribers for all Packages

Use this procedure to display the list of subscribers for all packages on your server.

> To display the subscribers for all packages

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Publishing**.

The server displays a list of all packages, their subscribers, and releases.

Adding Subscribers from a Publishing Server

When you add a subscriber, you are identifying the Integration Servers that are to receive a package. You can have a different list of subscribers for each package on your server.

Specify the subscribers (recipients) of the package. (You only need to execute this task the first time you publish the package; from then on, you can simply modify or reuse the initial list.)

Keep the following points in mind when adding a subscriber from a publishing server:

- To specify the automatic pull feature, you must create the subscription from the subscriber.
- The subscribing server must be running at the time you add the subscriber.

Note:

To request a subscription from a subscribing server, see [“Subscribing to a Package from a Subscribing Server”](#) on page 689.

> To add a subscriber

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Publishing**.
3. Click **Add Subscribers**.
4. Select the package for which you want to identify subscribers from the drop down list in the **Package** field.
5. To identify a subscribing server, enter information in the following fields:

Field	Description
Host Name	Name of the machine on which the subscribing server is running.

Field	Description
Host Port	Port number on which the subscribing server listens for this package to be published.
Transport	Method the publishing server uses to send the package to the subscribing server. Select HTTP or HTTPS. HTTP is the default. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If you want the publisher to use SSL when sending the package to the subscriber, you must specify HTTPS here.</p> </div> <p>When the publisher connects to the subscriber, the publisher uses the server's default Outbound SSL Certificates as specified on the publisher's Security > Certificates screen.</p>
Remote User Name	User the publishing server uses to log into the subscriber server. This user must be a member of a group that is assigned to the Replicators ACL on the subscribing server.
Remote Password	Password of the user that the publishing server uses to log into the subscribing server.
Notification E-mail	E-mail address of the administrator to notify when the publishing server releases a package.

6. Click **Add Subscriber**. The server adds the subscriber to the list in the **Subscribers** field.

Repeat this step for each server you want to identify as a subscriber to the package.

Updating Subscriber Information

Use this procedure to update information about a subscriber, such as the package name.

➤ To update subscriber information

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of in the Navigation panel, click **Publishing**.
3. Click **Update and Remove Subscribers**.
4. Locate this subscriber in the information list and click **Edit** in the **Update** column.
5. To change subscriber information, enter information in the appropriate fields below:

Field	Description
Packages	Packages to which the subscriber subscribes. You can change the subscription to be for another package. You can only select a package to which your server does not already subscribe because you cannot both publish and subscribe to the same package.
Host Name	Name of the machine on which the subscribing server is running.
Host Port	Port number on which the subscribing server listens for this package to be published. The number you specify must correspond to a port that already exists and is enabled on the subscribing server. In addition, the publishing server must have replicator access or higher.
Transport	Method the publishing server uses to send the package to the subscribing server. Select HTTP or HTTPS. HTTP is the default. The transport type must match the type defined for the host port on the subscribing server.
	<p>Note: If you want the publisher to use SSL when sending the package to the subscriber, you must specify HTTPS here.</p> <p>When the publisher connects to the subscriber, the publisher uses the server's default Outbound SSL Certificates as specified on the publisher's Security > Certificates screen.</p>
Remote User Name	User the publishing server uses to log into the subscriber server. This user must be a member of a group that is assigned to the Replicators ACL on the subscribing server.
Remote Password	Password of the user that the publishing server uses to log into the subscribing server.
Notification Email	E-mail address of the administrator to notify when the publishing server releases a package.

- Click **Submit Changes**. The server adds the subscriber to the list in the **Subscribers** field. The server updates the information on both the subscribing and publishing servers.

Removing Subscribers for a Package

Use this procedure to remove a subscriber from a package that you publish.

Note:

If a subscriber removes a subscription initiated by the publisher, the subscribing server removes the subscription from its subscriptions list, but the subscription is not immediately removed from the publisher's list. Instead, the next time the publishing server tries to send the package to the subscriber, the publisher is notified of the removal and then deletes the subscription from the publisher's list.

> To remove subscribers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of in the Navigation panel, click **Publishing**.
3. Click **Remove Subscribers**.
4. Locate the package for which you want to remove subscribers and check the box in the **Delete** field.

Note:

If the subscriber is running when you remove it from the subscriber list, the publisher tells the subscriber it has been removed. However, if the subscriber is not running, the subscriber will not know the subscription has been canceled. In this case, you should manually delete the subscription from the subscriber server later when it is available.

Publishing a Package

Publishing a package to other Integration Servers involves two tasks:

- **Creating a release.** To publish a package, your server creates a distribution file that contains the information for the package.

When you create the distribution file, you select what information to include in the file.

You can select all files to send, or just some. In addition, you can request a *full* release or a *patch* release. With a full release, the new package entirely replaces the old package on the subscriber's server. With a patch release, the files in the patch release replace the versions of those files in the target package; all other files in the target package remain intact. See "[Overview of Package Replication](#)" on page 673 for more information about how full and patch releases differ.

After you indicate the files to include in the release, the server places all the selected files into a single, compressed file (a zip file). It places the zip file in the *Integration Server_directory \instances \instance_name \replicate \outbound* directory. If the outbound directory already contains a zip file for this package, the server overwrites the existing file.

- **Sending the release.** After you create the release, you can send it to the subscribing servers.

A subscribing server receives the zip file containing the release in its inbound directory (*Integration Server_directory \instances \instance_name \replicate \inbound*). If a zip file for the package already exists in a subscribing server's inbound directory, the server overwrites it. The zip file remains in the inbound directory on the subscribing server until the administrator of that server installs the package.

A developer can set up the package to execute a service when you create the release. When you begin to create the release, this service executes before the list of files to be zipped is displayed. You can use this service to write state and configuration information for the package to a file. This file will be included with the other zipped files included in the release. For instructions on setting up replication services, see *webMethods Service Development Help*.

Important:

Before you can publish a package, you must specify the subscribers. For instructions, refer to [“Adding Subscribers from a Publishing Server”](#) on page 680.

Creating a Release

Use the following procedure to create a release for a package.

> To create a release

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Publishing**.
3. Click **Create and Delete Releases**.
4. Locate the package for which you want to create a release, and click **Create Release for *PackageName***.
5. The server displays a screen from which you specify the files you want to include in the release, the type of release (full or patch), and version information. See [“Specifying File and Version Information for a Release or Archive”](#) on page 684 for instructions on specifying this information.

Sending a Release

Use the following procedure to send a package release.

> To send the release

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel click **Publishing**.
3. Locate the release of the package you want to send under **Available Releases**, and click **Send Release**.

Specifying File and Version Information for a Release or Archive

When you archive a package or create a release, the server displays a screen from which you can specify the files you want to archive or release, the type of archive or release (full or patch), and version information.

Note:

You can also use the `watt.server.createPackage.ignorePattern` configuration parameter to specify the patterns of file names for the files that you do not want Integration Server to include when

publishing packages. For more information about this parameter, see [“Server Configuration Parameters” on page 1017](#).

Refer to [“Archiving a Package” on page 672](#) for detailed instructions on how to archive a package. Also, refer to [“Publishing a Package” on page 683](#) for detailed instructions on how to create and send a release.

» To specify file and version information for a released or archived package

1. Identify the files that you want to include in the release/archive.

If you want to include Do this

All files	In the Files to include section, select All files .
Most, but not all, of the files	In the Files available in <i>package</i> , section, select the files you do NOT want to include in the archive or release. In the Files to include section, select all except selected files . If the developer added package dependencies or startup, shutdown, or replication services to the package since the last archive or release was created, be sure to include the manifest.v3 file. Otherwise these services will not be available in the resultant package. See “Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710 for more information about startup, shutdown, and replication services.
Only a few of the files	In the Files available in <i>package</i> section, select the files you want to include in the archive or release. In the Files to include section, select Selected files . If the developer added package dependencies or startup, shutdown, or replication services to the package since the last archive or release was created, be sure to include the manifest.v3 file. Otherwise these services will not be available in the resultant package. See “Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710 for more information about startup, shutdown, and replication services.
Files with a similar path name	In the Files to include section, select Files specified by filter and enter a valid filter, for example *.java or *.class. or To include all files <i>except</i> those with a similar name, in the Files to include section, click All except files specified by filter and enter a valid filter, for example *.bak.

If you want to include Do this

If the developer added package dependencies or startup, shutdown, or replication services to the package since the last archive or release was created, be sure to include the manifest.v3 file. Otherwise these services will not be available in the resultant package. See [“Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710](#) for more information about startup, shutdown, and replication services.

You can specify the following special character to perform pattern matching.

Char	Description	Example
*	Matches any number of characters	*.java

- In the **Files to Delete from Target Packages** field, identify the files that you want to delete from the target package. Separate each entry with a semicolon (;). When the subscribing server installs the package, the server deletes these files from the target package.
- Specify package version information and description:

Field	Description
Archive/Release Type	<p>Full: All files in the package are written to the archive or release</p> <p>Patch: Selected files in the package are written to the archive or release. When the administrator on the target server installs a patch archive or release, the files contained in the patch archive or release replace the versions of those files in the target package; all other files in the target package remain intact.</p> <p>If the developer added package dependencies or startup, shutdown, or replication services to the package since the last archive or release was created, be sure to include the manifest.v3 file. Otherwise these services will not be available in the resultant package. See “Running Services When Packages Are Loaded, Unloaded, or Replicated” on page 710 for more information about startup, shutdown, and replication services.</p>
Archive/Release Name	A name you assign to the archive or release, for example Beta Release of WmExample Package.
Brief Description	A description you assign to the archive or release, for example “Dec release with patches to correct OrderProcess problem.”
Version	The version number you assign to the package you are archiving or releasing. This version might not be the same as the version of the

Field	Description
	<p>package itself. When a developer first creates a package, the version number is set to 1.0.</p> <p>For more information about the checking the Integration Server performs, see “Version Checking” on page 677.</p>
Build Number	<p>A number that a developer assigns to a package each time it is regenerated. For example, a developer might generate version 1.0 of the WmExample package 10 times, assigning build number 1,2,3...10.</p>
Patches Included	<p>A list of patches that have been applied to this release of the package. These are numbers that are meaningful to your installation, possibly obtained from your problem tracking system.</p>

- Specify subscriber settings:

Field	What It Means
webMethods Integration Server	<p>Version of the webMethods server that must be running on the target server.</p> <p>For more information about the version checking performed by the subscribing server, see “Version Checking” on page 677.</p>
Minimum Version of JVM	<p>Minimum version of the Java Virtual Machine (JVM) that the target Integration Server should be running when using this package. When the administrator installs the package, the server checks the version of the JVM it is running. If it is running a lower major version of the JVM, the server installs the package but does not activate it.</p> <p>For more information about the version checking performed by the subscribing server, see “Version Checking” on page 677.</p>

- Specify version of target package (for patch releases only).
- This is the version of the package the target server must be running. When the administrator installs the patch on the target server, the server checks to make sure the version of the target package is the same as the one specified here. If the target package is a different version, the server does not install the package. This restriction gives you greater control over how and where patches are applied. This is useful because patches are typically release dependent.

The Subscribing Server

This section describes the tasks the subscribing server performs when participating in package replication as the subscribing server.

Subscribers can retrieve packages manually or automatically. To retrieve a package manually, an administrator on the subscribing server views a list of available subscriptions and retrieves the desired package. When *automatic* pull is in effect, the subscribing server automatically pulls a package from the publisher when a new release becomes available.

For a package to be retrieved automatically, the subscriber must specify the automatic pull feature when setting up the subscription. When a new release becomes available, the publishing server sends a service-invocation e-mail to a designated e-mail server. The service-invocation e-mail contains a call to a service that runs on the subscribing server to retrieve packages. The subscribing server periodically checks the e-mail server through an e-mail port on the subscribing server. When it receives and processes the service-invocation e-mail, the subscribing server automatically pulls the package from the publisher and places it in the Inbound directory. The administrator on the subscribing server can then install the package.

Task	Refer to
Displaying packages to which your server subscribes	“Displaying Packages to Which Your Server Subscribes” on page 688
Manually Pulling a Package	“Manually Pulling a Package” on page 689
Subscribing to a package from another server	“Subscribing to a Package from a Subscribing Server” on page 689
Updating subscription information	“Updating Subscriber Information” on page 681
Canceling a subscription to a package on another server	“Canceling a Subscription” on page 694
Installing a package that was published from another server	“About Installing Packages Published by Another Server” on page 695

Displaying Packages to Which Your Server Subscribes

You can view the subscriptions your server has to packages on other servers.

➤ To display the packages to which your server subscribes

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel click **Subscribing**.
3. The server displays a list of your available subscriptions, organized by publisher, package, and release.
4. The server automatically updates this information when you (the subscriber) add, update, or remove a subscription; however, to see changes made by the publishers, you must click **Update All Subscription Details**.

Manually Pulling a Package

You can manually pull subscriptions to the inbound directory of your server.

➤ To pull a package to which have already subscribed

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Subscribing**.
3. The server displays a list of your available subscriptions, organized by publisher, package, and release.
4. Find the release of the package you want to pull and in the **Retrieve** field, click the retrieval method you want to use.

Field	What It Means
Via Service Invocation	The publishing server sends the release using HTTP or HTTPS.
Via FTP Download	<p>The publishing server sends the release using FTP.</p> <p>When you select FTP, the server prompts you for information required to use FTP:</p> <p>Release Name: Name assigned to the release, for example Beta Release of WmExample Package.</p> <p>Remote Server Alias: Name of the machine on which the publishing server resides.</p> <p>Remote Server FTP Port: FTP port on the publishing server through which the publisher will send the package.</p> <p>Remote User Name: User that the subscriber uses to log into the publishing server.</p> <p>Remote Password: Password of the user that the subscribing server uses to log into the subscribing server.</p>

5. Install the package. For instructions, see [“About Installing Packages Published by Another Server”](#) on page 695.

Subscribing to a Package from a Subscribing Server

When you subscribe to a package from the subscribing server, your server sends a subscription request to the publishing server. The publishing server adds your server to the subscription list for the package.

The remote server must have an alias defined on the local server. If the remote server does not already have an alias defined, you can define one ahead of time by going to the **Settings** menu of the Navigation panel and clicking **Remote Servers** or you can define one while creating the subscription.

When requesting a subscription, the subscriber must provide the following two-way connection information to the publisher:

- **Method the subscriber will use to connect to the publisher to make the subscription request.** The subscriber must supply a valid userid and password it can use to log on to the publishing server. You set up this and other connection information using a remote server alias for the publisher.
- **Method the publisher will use to connect to the subscriber when sending it a package.** The subscriber must supply a valid userid and password that the publisher can use to log on to the subscribing server. This userid must be a member of a group that is assigned to the Replicators ACL. In addition, the subscriber must supply other connection information, such as listening port.

Requesting a Subscription to a Package from Another Server

The following procedures describe how to request a subscription.

Note:

The publishing server must be running at the time you add the subscription

➤ **To request a subscription to a package from another server**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Subscribing**.
3. Click **Subscribe to Remote Package**.
4. Type the name of the package in the **Package** field. Be sure to type the name exactly as it is specified on the publishing server, using the same combination of upper- and lowercase characters.
5. Enter the information in the following fields to set up your request:

Field	Description
Publisher Alias	Alias assigned to the publisher. The alias definition tells the subscriber how to connect to the publishing server to register for a subscription. The alias contains connection information such as host name or IP address. If you have not already defined an alias for this publisher, click the link to go to the Remote Servers screen. From this screen you can set

Field	Description
	up an alias for the publisher. See “Setting Up Aliases for Remote Integration Servers” on page 115 for more information.
Local Port	Port number on which the subscriber listens for the publisher to send the package. This setting determines whether the publisher uses HTTP or HTTPS.
	<p>Important: If you want the publisher to use SSL when sending the package to the subscriber, you must specify an HTTPS port here.</p>
Local Password	Password for the local user name.
Notification Email	E-mail address to notify when the publishing server releases a package or a package is delivered.
Automatic Pull	<p>Specifies whether the subscribing server is to automatically pull the package from the publisher when a new release becomes available.</p> <p>If you select Yes, you must also specify the e-mail address of a user on an e-mail server to which the publishing server should send a service-invocation e-mail.</p> <p>The subscribing server, through an e-mail port, periodically checks this e-mail address for a service-invocation e-mail. When the subscribing server processes the e-mail, it pulls the package.</p> <p>The service invocation e-mail contains a call to a service that runs on the subscribing server and loads the package to the subscribing server's Inbound directory.</p> <p>For automatic pull to work, you must set up an e-mail port to listen at the automatic pull address (described below).</p> <p>For information about setting up an e-mail port, see “Adding an E-Mail Port” on page 181.</p>
Automatic Pull Email	<p>E-mail address to which the publishing server is to send a service-invocation e-mail when a new release of the package becomes available.</p> <p>Use a different e-mail address for the notification and service-invocation e-mails. For example, send notification e-mails to <code>package_notifications@mymailserver.com</code> and service invocation e-mails to <code>package_autopulls@mymailserver.com</code>.</p> <p>For automatic pull to work, you must set up an e-mail port to listen at this address.</p>

Field	Description
	For information about setting up an e-mail port, see “Adding an E-Mail Port” on page 181 .

6. Click **Start Subscription**.

Important:

If you request a subscription to a package that does not exist on the specified server, or if that server does not own the package (i.e., it is a subscriber of the package), you will receive an error message, and the publishing server does not process your subscription.

Updating Your Subscription Information

Use this procedure to update information about your subscription, such as the user name or password on the subscribing server.

➤ To update your subscription information

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of in the Navigation panel, click **Subscribing**.
3. Click **Update and Unsubscribe from Remote Package**.
4. Click **Edit** in the **Update** column for the package you want to update.
5. To change subscription information, enter information in the appropriate fields below:

Field	Description
Package	Package for which you want to change subscription information. You can change the package to another package if you do not already subscribe to or publish the new package. This restriction exists because you cannot both subscribe to and publish the <i>same</i> package.
Publisher Alias	Alias assigned to the publisher. The alias definition tells the subscriber how to connect to the publishing server to register for a subscription. The alias contains connection information such as host name or IP address. If you have not already defined an alias for this publisher, click the link to go to the Remote Servers screen. From this screen you can set up an alias for the publisher. See “Setting Up Aliases for Remote Integration Servers” on page 115 for more information.

Field	Description
Local Port	<p>Port number on which the subscriber listens for the publisher to send the package. This setting determines whether the publisher uses HTTP or HTTPS.</p> <p>Important: If you want the publisher to use SSL when sending the package to the subscriber, you must specify an HTTPS port here.</p> <p>Note: When the publisher connects to the subscriber, the publisher uses its default certificate (specified on its Security Settings screen). Make sure the port you specify here can accept that certificate.</p>
Local User Name	<p>User as which the publisher will log into the subscriber.</p> <p>This user must belong to a user group that is assigned to the Replicators ACL. If you delete the user or change its association with the Replicators ACL, the publisher cannot send this package to the subscriber.</p>
Local Password	<p>Password for the local user name.</p>
Notification E-mail	<p>E-mail address to notify when the publishing server releases a package or a package is delivered.</p>
Automatic Pull	<p>Specifies whether the subscribing server is to automatically pull the package from the publisher when a new release becomes available.</p> <p>If you already have automatic pull configured and want to turn it off, select No. Then go to the Automatic Pull E-mail field and delete the e-mail address there.</p> <p>If you want to configure your server for Automatic Pull, select Yes. You must also specify the e-mail address of a user on an e-mail server to which the publishing server should send a service-invocation e-mail.</p> <p>The subscribing server, through an e-mail port, periodically checks this e-mail address for a service-invocation e-mail. When the subscribing server processes the e-mail, it pulls the package.</p> <p>The service invocation-e-mail contains a call to a service that runs on the subscribing server and loads the package to the subscribing server's Inbound directory.</p> <p>For automatic pull to work, you must set up an e-mail port to listen at the automatic pull address (described below).</p> <p>For information about setting up an e-mail port, see “Adding an E-Mail Port” on page 181.</p>

Field	Description
Automatic Pull E-mail	<p>E-mail address to which the publishing server is to send a service-invocation e-mail when a new release of the package becomes available.</p> <p>Use a different e-mail address for the notification and service-invocation e-mails. For example, send notification e-mails to <code>package_notifications@mymailserver.com</code> and service invocation e-mails to <code>package_autopulls@mymailserver.com</code>.</p> <p>For automatic pull to work, you must set up an e-mail port to listen at this address.</p> <p>For information about setting up an e-mail port, see “Adding an E-Mail Port” on page 181.</p>

Note:

The publishing server must be running at the time you add the subscription.

6. Click **Submit Changes**.

The server updates the information on both the subscribing and publishing servers.

Canceling a Subscription

When you cancel a subscription, the server sends your cancellation notice to the publishing server. The publishing server removes your server from the subscription list for the specified package. If the publisher is not running when you cancel your subscription, the next time the publisher tries to send the package to your server, the publisher is informed of the cancellation and automatically deletes the subscription from its list of subscribers.

Note:

If a subscriber removes a subscription initiated by the publisher, the subscribing server removes the subscription from its subscriptions list, but the subscription is not immediately removed from the publisher's list. Instead, the next time the publishing server tries to send the package to the subscriber, the publisher is notified of the removal and then deletes the subscription from the publisher's list.

➤ To cancel your subscription to a package on another server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Subscribing**.
3. Click **Update and Unsubscribe from Remote Package**.
4. Locate the package for which you want to cancel the subscription and click the **✗** icon.

About Installing Packages Published by Another Server

When another server publishes a package to your server, you need to install the published package. You can also perform the following tasks on a package:

- **Recover a package.** If you install a package that has the same name as an existing package on your server, your server copies the original package to *Integration Server_directory \instances\instance_name\replicate\salvage* before it installs the new one. This lets you easily revert to the previous version if you are dissatisfied with the new package. For information about reverting to the earlier version, refer to [“Recovering a Package” on page 672](#).
- **Activate a package.** You can select whether you want the server to immediately activate the package after it installs it. If Integration Server does not activate a package upon installation, Integration Server copies the package to the packages directory, but it is not available for clients to use. To make this package available for clients, you must manually activate it. For more information, refer to [“Activating a Package” on page 668](#).
- **Archive a package.** You can also archive the package automatically after installing the package. If you choose to archive the package automatically, Integration Server moves the package from the *Integration Server_directory \instances\instance_name\replicate\inbound* directory to the *Integration Server_directory \instances\instance_name\replicate\archive* directory. If you do not want to archive a package automatically after installation, you can archive it later. For more information, see [“Archiving a Package” on page 672](#).

Installing a Package Published by Another Server

When installing a package published by another server, keep the following points in mind:

- A package coming from the publisher already has a List ACL associated with it, specifically, the List ACL that was assigned to the package on the publishing server.

The installing user does not need to be a member of that ACL to install the package; however, users on the subscribing server must be members of the package's List ACL in order to display the package on Integration Server Administrator screens.

Important:

Be sure the server where you are installing the package has an ACL with the same name as package's List ACL. If the List ACL does not exist, no users will be able to display the package on Integration Server Administrator screens.

- If the package you are installing has dependencies on another package that does not exist on your server, the server will install the package but will not enable it. You will not be able to enable the installed package until the dependencies are satisfied.
- If the package you are installing is associated with an e-mail listener, the server will install the package but will not enable the listener. This is because the password required for the Integration Server to connect to the e-mail server was not sent with other configuration information about the listener. To enable the listener, go to the **Security > Ports > Edit E-mail Client Configuration** Screen and update the **Password** field to specify the password needed to connect to the e-mail server.

- When you install a package from another Integration Server, it is possible that an HTTP URL alias associated with the new package has the same name as an HTTP URL alias already defined on your Integration Server. If Integration Server detects a duplicate alias name, it will display load warnings on the **Packages > Management > *package_name*** screen. For more information about duplicate HTTP URL aliases, refer to [“Portability of URL Aliases and Possible Conflicts”](#) on page 433.
- When you install a package from another Integration Server, it is possible that a port alias for a port in the new package has the same alias as a port already defined on your Integration Server. A port alias must be unique across the Integration Server. If Integration Server detects a duplicate port alias, it will display load warnings on the **Packages > Management > *package_name*** screen. Additionally, Integration Server will not create the port and will write the following warning to the server.log:

```
[ISS.0070.0030W] Duplicate alias duplicateAlias encountered creating protocol listener on port portNumber
```

Note:

If you want the port to be created when the package is loaded delete the existing port with that alias, create a new port that has the same properties as the just deleted port, and then reload the package containing the port with the duplicate alias. Integration Server creates the port when the package is reloaded.

- When you install a package released from a version of Integration Server prior to version 9.5, Integration Server creates an alias for each port associated with the package. For more information about the port alias created by Integration Server, see [“About the Port Alias”](#) on page 152.
- If hot deployment of packages is enabled, Integration Server installs the package using the hot deployment process. For more information about hot deployment, see [“Hot Deployment of Packages”](#) on page 697.

Important:

Make sure that packages you install come from a legitimate source, such as a replication from another server. If you are not sure, check with the developers in your organization to verify that an authorized person updated the package. Unknown packages might contain code that could damage your server.

> To install a package that was published from another server

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. Click **Install Inbound Releases**.
4. Select the package you want to install from the **Release file name** drop down list.

5. If you want Integration Server to make the package available immediately following installation, select the **Activate upon installation** check box in the **Option** field.
6. If you want Integration Server to archive the package automatically after installation, select the **Archive upon installation** check box. This check box is selected by default.
7. Click **Install Release**.

Integration Server installs the package and displays a message to that effect.

Using a Package Class Loader

By default, when a service in a package needs to load a class into storage, the package's class loader defers the search for the class to its parent class loader, which is always the Integration Server class loader. However, there may be times when you want Integration Server to use the package's class loader instead. For example, you might want to use a newer version of a class that exists in the package directory, rather than the version the Integration Server class loader will access.

For the package class loader to have access to class you want to load, the jar files containing the class you want load must exist in this directory:

Integration Server_directory \instances*instance_name*\packages*packageName*\code\jars directory.

» To configure Integration Server to use a package class loader

1. Navigate to the *Integration Server_directory* \instances*instance_name*\packages*packageName* directory, where *packageName* is the name of the package that has the jar files that you want to load.
2. Add the following property to the manifest.v3 file:

```
<value name='classloader'>package</value>
```

3. Reload the package.

For more information about how Integration Server loads classes, refer to [“Class Loading in Integration Server” on page 911](#)

Hot Deployment of Packages

Hot deployment of packages is the process of seamlessly installing and upgrading custom packages in Integration Server in a production environment while ensuring that Integration Server assets are available for processing without any noticeable downtime. Integration Server supports hot deployment of packages using Integration Server Administrator as well as Deployer.

When you deploy a new version of a package, Integration Server unloads the existing package information from memory and loads the new version of the package and its contents into memory. While the package is being upgraded, it is possible that some assets might be unavailable for some

time. This could cause serious issues such as denial of service or disruption of the in-flight tasks. Hot deployment of packages ensures that the packages are available at all times, thus, ensuring that there is no downtime, which is crucial in a production environment.

Traditionally, to upgrade the packages in an Integration Server cluster without shutting down the entire cluster, a rolling quiesce and upgrade process is used. During the rolling upgrade of a cluster, each Integration Server in the cluster is individually quiesced, upgraded, and restarted while the other servers in the cluster continue to process the requests. The limitation with this approach is that while Integration Server is quiesced, all the external ports are disabled and Integration Servers will not be in a state to process requests or execute services. Hot deployment of packages makes it easier to deploy packages without having to quiesce each Integration Server. In case of hot deployment, only the assets that are part of or dependent on the packages being deployed are affected and not the entire server. However, quiesce mode in Integration Server is essential to perform tasks such as installing fixes and performing other maintenance tasks.

How Hot Deployment Works

When you configure Integration Server to perform hot deployment of packages, Integration Server employs a block and wait approach. That is, Integration Server waits for the in-flight tasks that are related to the packages that are being deployed to complete before unloading the existing package information from memory. At the same time, when the hot deployment of packages starts, Integration Server blocks access to the assets that belong to or have dependency on the packages that are being deployed.

Integration Server ensures that there are no failures in the processing of assets by taking the following steps:

- Wait for all in-flight tasks that are dependent on the packages being deployed to complete. You can specify the period of time for which Integration Server must wait for the in-flight tasks to complete before attempting to deploy packages. If the tasks are not done within the specified timeout period, the hot deployment fails. Integration Server also provides the option to recover the original packages in case hot deployment fails, thus ensuring there is no downtime.
- Block the execution of assets that belong to or have dependency on the packages that are being deployed. For example, if a service that is being invoked belongs to a package that is getting deployed, Integration Server will block the service invocation until the package is upgraded. When determining which assets have a dependency, Integration Server considers only those assets for which you have explicitly specified the package dependencies.

Determining Package Dependencies during Hot Deployment

To ensure that there is no disruption of in-flight tasks, it is important that you set the package dependencies correctly. During hot deployment of packages, Integration Server employs the block and wait approach on all assets in the packages that are being deployed as well as all its identified dependents. If you do not set the dependencies, Integration Server might go ahead with the invocation of an asset. If the asset is being upgraded at that time, the invocation could fail with an error.

For example, if you have not explicitly specified “FinanceUtil” as a dependent package of the “Finance” package, Integration Server will not wait for the completion of the execution of an asset in “FinanceUtil” package before deploying the “Finance” package. This could result in the hot deployment of the “Finance” package failing. Integration Server will wait for the completion of the execution of an asset in “FinanceUtil” package before hot deploying the “Finance” package only if you explicitly specify “FinanceUtil” as a dependent package of the “Finance” package. For more information about package dependencies, see *webMethods Service Development Help*.

Considerations for Hot Deployment of Packages

Before you configure or preform hot deployment, keep the following behavior, limitations, and considerations in mind:

- When enabled, Integration Server performs hot deployment for the installation of any custom package regardless of whether it is a new package or an upgraded package.
- You can perform hot deployment for the entire package only. You cannot perform hot deployment for specific assets in a package. During hot deployment, Integration Server upgrades all the assets in a package.
- To perform hot deployment of packages, you do not have to quiesce or restart Integration Server nodes.
- Hot deployment is supported only for the assets that are owned by the Integration Server. For example, Flow Service and Java Service.
- Integration Server ensures that processes such as HTTP/HTTPS/FTP/FTPS requests, scheduler tasks, file polling requests, emails, Messaging/JMS requests, and web service requests continue while performing hot deployment of packages. During deployment, there might be a delay in response from those assets that are affected by the hot deployment of packages because Integration Server blocks the execution of assets that belong to or have dependency on the packages that are being deployed.

For example, during hot deployment, when Integration Server unloads the existing package information from memory, Integration Server disables the ports associated with the deployed packages. These ports will be enabled when the packages are activated successfully. Therefore, if you are hot deploying a package that has a port associated with it, access to services or other resources through those ports will fail until the port is successfully enabled.

To avoid the situation where a port in the package undergoing hot deployment is disabled, assign ports to packages that are not deployed using hot deployment.

Note:

When you create a port, you assign the port to a package. Integration Server stores the listener configuration information for the port in the package that you specify. Enabling or disabling the package that contains a port, enables or disables the port as well.

- If any of the assets in the package being deployed or in dependent packages are currently being processed at the time of deployment, Integration Server waits for the processing to complete or waits a specified timeout period, whichever is earlier, before attempting to deploy the package. If the tasks are not done within the specified timeout period, the hot deployment

might fail. To help ensure that hot deployment is successful, you must configure the hot deployment timeout value to a value that is greater than the time taken for the task to complete.

With regards to scheduled user tasks, Integration Server assesses the dependency of packages that are being deployed and permits the completion of any scheduled user tasks that are already running. Until hot deployment completes, Integration Server suspends any scheduled user tasks that have not yet started.

- Hot deployment is not cluster aware. Hot deployment of packages may not happen at the same time on all the server nodes in a cluster. Also, info and debug messages related to hot deployment in one server node will not be shared across the cluster nodes.
- In case the deployment of a new version of package fails, Integration Server provides the auto recover configuration option that you can use to instruct Integration Server to recover the earlier version of the package automatically.

Note:

When you install a package for the first time in Integration Server, if there is any error in hot deployment, Integration Server will not be able to automatically recover the package because a working copy of the package does not exist.

- You can use the `pub.packages.hotdeployment:cancel` service to cancel a hot deployment operation. For more information, see *webMethods Integration Server Built-In Services Reference* .

Enabling and Configuring Hot Deployment of Custom Packages

When you enable hot deployment, Integration Server performs hot deployment when installing any new custom package or any upgraded version of a custom package. Before configuring hot deployment for Integration Server, make sure to review the information in [“Determining Package Dependencies during Hot Deployment” on page 698](#) and [“Considerations for Hot Deployment of Packages” on page 699](#).

➤ To enable and configure hot deployment of custom packages

1. In Integration Server Administrator, click **Settings > Hot Deployment > Edit Hot Deployment Settings**.
2. Next to **Enable**, select **Yes** to enable hot deployment of packages.
3. In the **Timeout** field, specify the maximum time in seconds Integration Server should wait for the in-flight processing to complete before attempting to hot deploy the packages. If there are any long running tasks that you want to complete before deploying the packages, make sure that you set the timeout value to a value more than the time it takes for the task to complete. The default is 60 seconds.
4. Select **Yes** for the **Auto Recover** option, if you want Integration Server to revert to the previous version of the package in case the deployment of a new version of the package fails. When you select **Yes** for the **Auto Recover** option, Integration Server copies the original package

to the *Integration Server_directory \instances \instance_name \replicate \salvage* directory before it deploys the new version. Thus, in case of any error during hot deployment, Integration Server recovers and reverts to the original package.

5. Click **Save Changes**.

Disabling Hot Deployment of Custom Packages

Use the following procedure to disable hot deployment for the installation of custom packages.

➤ To disable hot deployment of custom packages

1. In Integration Server Administrator, click **Settings > Hot Deployment > Edit Hot Deployment Settings**.
2. Next to **Enable**, select **No** to disable hot deployment of packages.
3. Click **Save Changes**.

Automatic Package Deployment

Automatic package deployment allows Microservices Runtime to install or upgrade packages automatically. When automatic package deployment is enabled, packages can be installed or upgraded without using Deployer or an administrator tool such as, Microservices Runtime Administrator. Automatic deployment can be used with an on-premises Microservices Runtime and an Microservices Runtime that runs inside a Docker container.

Automatically deploying packages can be particularly useful for a Microservices Runtime that runs in a Docker container. Packages to be installed or updated on the container can be placed in an `autodeploy` directory that Microservices Runtime periodically scans. Microservices Runtime will find and install the new or updated packages. If you do not use automatic deployment, you must rebuild the Docker image to use any new or updated packages with Microservices Runtime container. For more information, see *Developing Microservices with webMethods Microservices Runtime*.

Note:

The automatic package deployment feature is available by default for a webMethods Microservices Runtime. To use the automatic package deployment feature with Integration Server, your Integration Server must have additional licensing.

36 Managing Services

■ About Services	704
■ Fully Qualified Service Names	704
■ Finding Information about Services and Folders	705
■ Manually Adding a Service to the Server	707
■ Testing Services	707
■ Canceling and Killing Threads Associated with a Service	708
■ Running Services When Packages Are Loaded, Unloaded, or Replicated	710
■ Running Services in Response to Specific Events	712
■ Managing Global Variables	712

About Services

A service is a server-resident unit of functionality that clients can invoke. A service might be an entire application or used as part of a larger application. There are several types of services: flow services (including web service connectors), adapter services, Java services, and C/C++ services.

You can create all flow services using Software AG Designer. You can create database flow services from the Integration Server Administrator as well. You can also use Designer to create adapter services, Java services, or use your own development environment to create Java and C/C++ services. For more information about the types of services and how to create them, see *webMethods Service Development Help*.

You can designate one or more services in a package as a startup, shutdown, or replication service. A startup service is a service that the server automatically executes when a package is loaded. A shutdown service is a service that the server automatically executes when a package is unloaded. A replication service is a service that the server automatically executes when a package is replicated.

To improve the performance of services, you can have the server cache the service results. Then, when the server receives subsequent requests for the service, it returns the cached results rather than executing the service. For more information, see [“Caching Service Results” on page 733](#).

You can use the scheduling function of Integration Server to schedule services to execute at times you specify. For more information on scheduling services, see [“Scheduling Services” on page 715](#).

Fully Qualified Service Names

The fully qualified service name is comprised of two parts: a *folder identifier* and the *service name*. The folder identifier consists of one or more folder names. The service name is a single name of the service.

Use a folder to group related services together. When a folder contains other folders, the nested folders are called subfolders. For example, if you have several services that involve financial information, you might create a folder named "Finance" to hold the services. Within the financial services, there might be services that are for personal finances. You might create a subfolder named "Personal" to hold those services.

Use any name for the service name. For example, if one of the financial services obtains stock quotes, you might name the service, "StockQuote."

To specify a fully qualified service name, identify the folder portion, then a colon (:), then the service name:

folder:service

For example, if the "StockQuote" service is in the "Finance" folder, the fully qualified service name is:

Finance:StockQuote

If the folder portion identifies more than one folder, separate each folder name with a period.

folder.subfolder1.subfolder2:service

For example, if the "HomeLoan" service is in the "Personal" folder, which is contained in the "Finance" folder, the fully qualified service name is:

Finance.Personal:HomeLoan

The *fully qualified name of each service must be unique within the server*. In addition, the fully qualified name of a service cannot be the same as the fully qualified name of any specification or document type that resides on the server.

Note:

The `watt.server.illegalNSChars` setting in the `server.cnf` file (which is located in the `IntegrationServer_directory\instances\instance_name\config` directory) defines the characters that you cannot use when naming folders and services. To view or change this setting, use the **Settings > Extended** screen from the Integration Server Administrator as described in page ["Working with Extended Configuration Settings" on page 127](#).

Package Names and Service Names

The relationship between the package name and the folder name can cause confusion. The name of the package to which a service belongs has no bearing on the names of the services and folders it contains. Nor does it affect how it is referenced by a client application. For example, if you move a service called "Personnel:GetDeptNames" from a package called "Admin" to a package called "EmployeeData" you will not affect client applications that reference that service; it will still be referenced by the name "Personnel:GetDepNames."

Because the fully qualified name of each service must be unique within the server, you cannot have two identically named services in two different packages on the same server.

HTTP URL Aliases for Services

You can create aliases for the HTTP URLs that users specify to invoke services. Aliases are easier to type than full URL path names, make it easier to move resources on Integration Server, and are more secure than URL path names, which allow users to see directory and file names.

You can create an HTTP URL alias from Designer or Integration Server. For more information, see ["Setting Up HTTP URL Aliases" on page 423](#).

Finding Information about Services and Folders

This section describes how to list the services (and folders) on your server and display information about a specific service.

Listing Folders and Services

The **Folders and Services** screens of the Integration Server Administrator list the services that reside on your server and the folders with which they are associated.

➤ **To list folders and services**

1. Open the Integration Server Administrator if it is not already open.
2. From the **Packages** menu in the Navigation panel, click **Management**.
3. Click **Browse Folders**.
4. To view the contents of a folder, click the folder name. The server displays another **Folder List** screen. For the selected folder, the server displays the subfolders followed by the services.

You can continue to click on folder names to view subfolders and services in selected folders.

Note:

The server will only display folders and elements to which you have List access.

Displaying Information about a Service

The **Packages > Management > WmPublic > Services > service** screen displays a variety of information about a selected service or specification.

> To display information about a service

1. Open the Integration Server Administrator if it is not already open.
2. From the **Packages** menu in the Navigation panel, click **Management**.
3. From the **Package List**, click the package whose services you want to view.
4. Click **Browse Services in** *packagename*.
5. Click the name of the service or specification for which you want to display information. For a description of the information that Integration Server Administrator displays for a service, see [“Service Information” on page 706](#).

Service Information

Integration Server Administrator displays the following information for a service.

Section	Description
General Info	<p>Identifies</p> <ul style="list-style-type: none"> ■ The folder in which the service is contained and the service name. ■ The name of the package with which the service is associated. ■ The type of service: Flow, Java, or C/C++.

Section	Description
	<ul style="list-style-type: none"> ■ Whether or not the service is stateless. ■ The allowed HTTP methods.
Universal Name	<p>The name that will be used to qualify the name of this service. It consists of the namespace name and the local name.</p> <p>By convention, a URI is generally used as the namespace name (e.g., http://www.gsx.com/gl). This assures that the universal name is globally unique.</p> <p>The local name uniquely identifies the service within the collection encompassed by namespace name. Most sites use the unqualified portion of the service name as the local name</p> <p>You may use any sequence of characters or digits for the namespace name and the local name.</p>
Java-Specific Parameters	For a Java service, identifies the Java class name and method name for the service.
Access Control	Identifies the ACLs assigned to the service or specification. For information about ACLs, services, and specifications, see “Controlling Access to Resources with ACLs” on page 501 .
Cache Control	Identifies whether the server is to save the results of executing this service in cache. For information, see “Caching Service Results” on page 733 .
Data Formatting	Identifies the name of a binding service that parses incoming XML for the service, the output template associated with the service (if any), and the type of the output template (HTML or XML). For information about output templates, refer to the <i>Dynamic Server Pages and Output Templates Developer’s Guide</i> .

Manually Adding a Service to the Server

If you have Java or C/C++ services that were not created using Designer or Developer, you must manually add them to the server using the jcode utility. For more information, see information about building Java services with your own IDE in *webMethods Service Development Help*.

Testing Services

You can use Integration Server Administrator to test the operation of a service. This allows you to quickly and easily verify the operation of a service and test it with special-case input values.

Note: Integration Server Administrator provides a limited testing environment as you can only specify input values for variables of type String. Designer offers a more robust environment for testing services.

> To test a service

1. Open the Integration Server Administrator if it is not already open.
2. From the **Packages** menu in the Navigation panel, click **Management**.
3. From the **Package List**, click the package whose service you want to test.
4. Click **Browse Services in *packagename*** .
5. Click on the name of the service you want to test.
6. To test the service, click **Test *servicename*** .

The server displays the **Test *ServiceName*** screen.

7. If you want to test the service with input values, fill in the required input information in the **Assign Input Values** section of the screen and click **Test with inputs**.

If you want to test the service without specifying input values, click **Test (without inputs)**.

Canceling and Killing Threads Associated with a Service

If you suspect that a flow service or Java service is unresponsive because it is waiting for an external resource or is in an infinite loop, you can stop execution of one or more threads associated with the service. You have two options:

- **Cancel the thread.** When you cancel a thread, Integration Server frees up resources that are held by the thread. For example, if a thread is holding a JDBC connection, Integration Server closes and releases the connection and returns it to the JDBC connection pool.
- **Kill the thread.** When you kill a thread, Integration Server attempts to free up resources, but if it cannot, it continues and terminates the thread. Integration Server replenishes the pool with a new thread.

Integration Server requires you to try canceling a thread before it will allow you to kill a thread. Once you successfully cancel a thread, you no longer have the option of killing it.

When you cancel or kill a thread, Integration Server returns a thread to the global, trigger, scheduler, or circuit breaker thread pool, depending on the function being performed.

Integration Server identifies which threads you can cancel or kill by flagging them on the **System Threads** screen as follows:

- Threads that can be canceled are marked with a ✓ in the **Cancel** column.
- Threads that can be killed are marked with an ✗ in the **Kill** column.

The ability to cancel or kill service threads is controlled by the `watt.server.threadKill.enabled` parameter. If you want to be able to cancel or kill threads, this parameter must be set to `true`, which is the default setting.

Note:

If you want Integration Server to interrupt a service that uses a canceled thread, you must set the `watt.server.threadKill.interruptThread.enabled` parameter to `true`.

Canceling or Killing a Thread

Before canceling or killing a thread, keep the following information in mind:

- If you want to be able to cancel or kill threads, the `watt.server.threadKill.enabled` parameter must be set to `true`, which is the default setting.
- Integration Server requires you to try canceling a thread before it will allow you to kill a thread.
- You cannot cancel or kill threads when a JDBC connection is waiting on a response from the database.
- You cannot cancel or kill threads when an underlying service is waiting on a blocking IO. Examples of blocking IO are:
 - A service waiting on a response from an HTTP request.
 - A service waiting on a response from an FTP request.
 - A service waiting on receiving data from read/write requests on a file or socket.
- You can cancel or kill threads for user-written Java or flow services only.
- You cannot cancel or kill threads for Integration Server system services.

CAUTION:

Use care when canceling or killing threads. Canceling a thread might not free up resources being held by the service. Killing a thread might put your resources in an unstable state.

➤ To cancel or kill a thread associated with a service

1. Open Integration Server Administrator if it is not already open.
2. From the **Server** menu in the Navigation panel, click **Statistics**.
3. In the **System Threads** field, in the **Current** column, click the number of current threads.

Integration Server displays the **System Threads** screen.

4. To make it easier to locate threads that can be canceled, select the **Show threads that can be canceled or killed at the top of the list** check box.

Integration Server dynamically updates the display to show threads that can be canceled (✓ in the **Cancel** column) or killed (✗ in the **Killed** column) at the top of the screen.

5. To display information about a thread, in the **Name** column for that thread, click the thread name.

The server displays a dump of that thread. Using the information provided in the thread dump, determine whether you want to cancel the thread.

To return to the **System Threads** screen, click **Return to System Threads**.

6. In the row for the thread you want to cancel, click the ✓ in the **Cancel** column.

Integration Server prompts you to confirm your action.

If the cancel is successful...

Integration Server removes the thread from the display.

If the cancel is not successful...

Integration Server updates the display to show an ✗ in the **Kill** column.

If you want to kill the thread, click the ✗.

Integration Server prompts you to confirm your action.

If Integration Server is able to kill the thread, it removes the thread from the display.

If Integration Server is not able to kill the thread, the thread remains in the display.

Running Services When Packages Are Loaded, Unloaded, or Replicated

To have the server automatically execute a prescribed set of operations each time the server loads or unloads a package from memory or replicates a package, you can identify startup, shutdown, and replication services. This section provides an overview of startup, shutdown, and replication services.

To identify these services you must use Designer. For instructions, see *webMethods Service Development Help*.

What Is a Startup Service?

A startup service is one that the server automatically executes when it loads a package. The server loads a package:

- At server initialization (if the package is enabled)
- When someone uses the Integration Server Administrator to reload a package

- When someone uses the Integration Server Administrator to enable a package

Startup services are useful for generating initialization files or assessing and preparing (e.g., setting up or cleaning up) the environment before the server loads a package. However, you can use a startup service for any purpose. For example, you might want to execute a time-consuming service at startup so that its cached result is immediately available to client applications.


What Is a Shutdown Service?

A shutdown service is one that the server automatically executes when it unloads a package from memory. If a package is in memory, the server unloads the package:

- At server shutdown or restart
- When someone uses the Integration Server Administrator to disable the package
- Before the server removes the package from memory when someone uses the Integration Server Administrator to reload a package

Shutdown services are useful for executing cleanup tasks such as closing files and purging temporary data. You could also use them to capture work-in-progress or state information before a package unloads.

What Is a Replication Service?

A replication service is one that the server automatically executes when it prepares to release or archive a package. The service executes when the administrator clicks the **Create Release** link on the **Packages > Publishing > Create and Delete Releases** screen or the **Archive** icon  on the **Packages > Management** screen.

Replication services provide a way for a package to persist state or configuration information so that this is available when the package is activated on the remote server.

Guidelines for Using Startup, Shutdown, and Replication Services

Keep the following guidelines in mind when using startup, shutdown, and replication services.

- When you create a startup or shutdown service, *you must register that service in the package with which it will be used*. When you create a replication service, you can register any valid service from any loaded package on the server, including the current package itself.
- Because services in a package are not made available to clients until that package's startup services finish executing, you should avoid implementing startup services that access busy remote servers. They will delay the availability of other services in that package.
- You may assign one or more startup services to a package; however, you cannot specify the order in which they will execute. If you have a series of operations that must execute in a specific order, encode the entire sequence within a single service or have a startup service invoke others.

See *webMethods Service Development Help* for instructions.

Running Services in Response to Specific Events

The Event Manager runs on the server, monitoring it for *events*. An event is a specific action that the Event Manager recognizes and an *event handler* can react to. An event handler is a service a developer writes to perform an action when a specific event occurs. The Event Manager recognizes a number of different events. For example, an *alarm event* occurs when the webMethods Integration Server throws an exception regarding the status or health of the server. The server generates alarm events when a user cannot log on to the server, a port cannot be started, a user is denied access to a port, and so on.

Developers control the Event Manager through Designer. The server saves information for event types and event subscriptions in the `eventcfg.bin` file. This file is generated the first time you start an Integration Server and is located in the following directory:

`IntegrationServer_directory\instances\instance_name\config`. There is no need for you to work with this file directly; however, if you are clustering Integration Servers, you need to copy this file from one server to another to duplicate event subscriptions on all servers in the cluster. For more information about using the Event Manager, see *webMethods Service Development Help*.

Managing Global Variables

Global variables are key/value pairs that you can use in flow services. At design time, instead of hard coding the values for variables, you can instruct Integration Server to use global variables.

To use global variable substitution in flow services, when assigning a value to a pipeline variable, you must specify the global variable as the value. You must also indicate that Integration Server should perform variable substitution. Then, at run time Integration Server substitutes the variables in the flow service with the values assigned to the respective global variables.

For example, you can define global variables for input parameters, such as `serverhost` or `serverport`, of `pub.client*` services. At design time, instruct Integration Server to use global variables defined for `serverhost` or `serverport` as the value of the corresponding variables in the pipeline. Then, at run time, Integration Server will substitute the variables in the flow service with the values assigned to `serverhost` or `serverport` global variables.

Using global variables makes it easy to change the value assigned to a pipeline variable. Instead of changing each mapping or flow service, you can change the value assigned to a variable by changing a single global variable.

For more information about global variable substitution, see *webMethods Service Development Help*.

Creating a Global Variable

You can define global variables using Integration Server Administrator. Integration Server saves the global variable definitions in `Integration Server_directory\instances\instance_name\config\globalVariables.cnf`.

When you create global variables, keep the following points in mind:

- Global variables can only be used to specify values for variables of type String, String List, and String Table.
- Global variable names:
 - Cannot include any special characters except period (.) and underscore (_).
 - Can be of a maximum length of 255 characters.
 - Are case-sensitive.
 - Must be unique on the Integration Server.
- If the global variable you are creating is a password, Integration Server encrypts and saves it using OPE (outbound password encryption).
- Global variable value can be of a maximum length of 255 characters.
- When editing a global variable, you can modify only the **Value** field.
- Global variables must be identical on all Integration Servers in a cluster.
- You can deploy global variables by using webMethods Deployer. For information about using Deployer, see the *webMethods Deployer User's Guide*.
- You can use global variable substitution when debugging a service.

> To create a global variable

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Global Variables**.
3. Click **Add Global Variable**.
4. Under **Global Variable** provide the following information:

Parameter	Specify
Is Password?	<p>Whether the global variable you are defining is a password.</p> <p>If the Is Password check box is selected, the text that you enter in the Value field is entered as a password, with asterisks indicating the input instead of characters. Integration Server encrypts and saves the text that you enter in the Value field as an outbound password.</p>
Key	A name for the global variable. Integration Server uses this key to refer to the global variable while performing global variable substitution.
Value	Value for the global variable.

5. Click **Save Changes**.

Deleting Global Variables

You can delete global variable definitions from Integration Server using Integration Server Administrator. If you delete a global variable that is used by a flow service, while executing the flow service, Integration Server will throw an exception and stop the service execution.

> To delete a global variable

1. Open Integration Server Administrator.
2. In the Navigation panel, select **Settings > Global Variables**.
3. Locate the row that contains the key/value pair that you want to delete, and click the **×** icon.
Integration Server prompts you to confirm that you want to delete the global variable.
4. Click **OK**.

37 Scheduling Services

■ Overview	716
■ Scheduling a User Task	716
■ Viewing Scheduled User Tasks	721
■ Updating Scheduled User Tasks	722
■ Suspending User Tasks	723
■ Resuming Suspended User Tasks	724
■ Canceling a Scheduled User Task	725
■ Viewing the Scheduled System Tasks	725
■ Simple Repeating Option	726
■ Complex Repeating Option	727
■ Target Node Options	730
■ How Transitioning to or from Daylight Savings Time Affects Scheduled Tasks	732

Overview

Use the server's scheduling function to schedule services to execute at times you specify. Services that you schedule are referred to as *user tasks*. You can:

- Create user tasks that Integration Server executes:
 - A single time
 - Repeatedly at a simple interval, for example hourly every day
 - Repeatedly at a complex interval, for example every other Tuesday in June
- View a list of scheduled tasks.
- Update scheduling options for existing user tasks.
- Cancel a scheduled user task before Integration Server completes all scheduled executions.
- Temporarily suspend a task's execution.
- Create user tasks that run on one, any, or all Integration Servers connected to the same database.
- Specify an action Integration Server is to take if a task has missed its scheduled execution time.

Note:

In addition to using Integration Server Administrator to schedule tasks, you can also perform scheduling by using a set of built-in services. See the *webMethods Integration Server Built-In Services Reference* for more information.

Tasks Provided by Integration Server

Integration Server provides user tasks that you can modify. For example, Integration Server supplies the Message History Sweeper task if you configured a document history database for exactly-once processing. This task removes expired entries from the document history database. Even though Integration Server scheduled this task, you can modify how often the service runs.

In addition to the scheduled user tasks that you set up, Integration Server schedules *system tasks* that it performs for normal system operation. You can view, but not update or cancel, most scheduled system tasks.


Scheduling a User Task

Use the following procedure to schedule a user task.

➤ **To schedule a user task**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.

3. Click **Create a Scheduled Task**.
4. Set the **Service Information** parameters as follows:

For this parameter	Specify
Description	A description of the task.
folder.subfolder:service	<p>The fully qualified service name of the service you want Integration Server to execute.</p> <p>Click Assign Inputs to enter new input values or modify existing values for the service you want Integration Server to execute.</p> <p>Before assigning input values, note that:</p> <ul style="list-style-type: none"> ■ You can only assign values for top-level input parameters of data type String. ■ Assigning input values using Assign Inputs overwrites any input values assigned to a service using the built-in services in thepub.schedulerfolder. <p>For information about specifying service names, see “Fully Qualified Service Names” on page 704.</p>
Run As User	<p>The user name you want the server to use when running the service. Click  to search for and select your user. A user can be selected from the local or central directory.</p> <p>Integration Server runs the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.</p>
Target Node	<p>Whether you want the task to run on other Integration Servers connected to the same database:</p> <ul style="list-style-type: none"> ■ Select Any server if the task needs to run on only one server connected to the database, and it does not matter which one. ■ For clustered servers only. Select All servers if the task needs to run on all Integration Servers in the cluster. ■ Select the name from the list of Integration Servers connected to the database if the task needs to run on only a specific server. <p>The default is the current Integration Server.</p>

For this parameter**Specify**

For more information about the **Target Node** options, see [“Target Node Options” on page 730](#).

5. Select an action to perform **If the Task Is Overdue**.

This parameter Specifies that Integration Server should

Run immediately Run the task immediately, no matter how late the task is.

Skip and run at next scheduled time Skip this execution of the task, and run it again at the next scheduled run time. This option is not available for tasks that run just once.

Suspend Place the task in a suspended state until an administrator resumes or cancels the task.

Integration Server periodically checks the status of scheduled tasks. If it finds a task that should have started but has not, Integration Server runs the task immediately, unless you have specified a special action to take for late tasks. Integration Server performs this "late action" if the task has missed its scheduled start time by a number of minutes you specify in the **if more than xxx minutes late** field.

Note:

The maximum number of minutes that the **if more than xxx minutes late** field can accept is 35000.

For tasks that are late but do not exceed the specified period, Integration Server runs the task immediately.

6. Select **Run Once**, **Repeating**, or **Complex Repeating** to indicate when and how often you want Integration Server to execute the service.

Select**To****Run Once**

Schedule a task to run just once. Specify the following:

- **Date.** The date on which you want Integration Server to execute the service.

Enter the date using the format YYYY/MM/DD. For example, if you want the server to execute the service on March 11, 2010, specify 2010/03/11.

- **Time.** The time at which you want Integration Server to execute the service.

Select	To
	<p>Enter the time using the format HH:MM:SS (using a 24-hour clock). For example, if you want Integration Server to execute the service at 1:00:00 a.m., specify 1:00:00; if you want Integration Server to execute the service at 1:00:00 p.m., specify 13:00:00.</p>
<p>Repeating</p>	<p>Schedule simple repeating tasks, for example once a day at a specific time. Specify the following</p> <ul style="list-style-type: none"> <p>■ Start Date and Start Time. The date and time of the first execution.</p> <p>For Start Date, use the format YYYY/MM/DD. For Start Time, use the format HH:MM:SS (using a 24-hour clock).</p> <p>For example, if you want the service executions to start on May 3, 2010 at 1:00:00 p.m., specify 2010/05/03 for Start Date and 13:00:00 for Start Time.</p> <p>■ End Date and End Time. The date and time of the last execution.</p> <p>For End Date, use the format YYYY/MM/DD. For the End Time, use the format HH:MM:SS (using a 24-hour clock).</p> <p>For example, if you want the service executions to stop on June 4, 2010 at 2:00:00 a.m., specify 2010/06/04 for End Date and 02:00:00 for End Time. Omitting End Date indicates that you want this service to execute for an indefinite period of time. If you omit End Time, Integration Server uses the current time.</p> <p>■ Interval. Execution interval.</p> <p>Enter the number of seconds that you want Integration Server to wait between executions of the service.</p> <p>■ Repeat after completion. Whether to wait for the previous execution of the service to complete before starting the next.</p> <p>For example, suppose the GetData service is scheduled to run every minute, but sometimes takes longer than that to complete. By default, Integration Server will start the next execution even though the previous one has not yet completed.</p> <p>If you check the Repeat after completion box, Integration Server will wait for the service to complete before running the next execution of the service. Executions that could not run while the service was executing are delayed.</p>

Select**To****Complex Repeating**

For more information about using the Repeating option, see [“Simple Repeating Option” on page 726](#).

Schedule a task that repeats at a more complex interval, for example, every other day, or twice a month. Specify the following:

- **Start Date** and **Start Time**. The date and time of the first execution.

For **Start Date**, use the format YYYY/MM/DD. For **Start Time**, use the format HH:MM:SS (using a 24-hour clock).

For example, if you want the service executions to start on May 3, 2010 at 1:00:00 p.m., specify 2010/05/03 for **Start Date** and 13:00:00 for **Start Time**. If you omit the **Start Date**, the first execution occurs on the first date as indicated by the **Run Mask** parameters. If you omit **Start Time**, the server uses the current time.

- **End Date** and **End Time**. The date and time of the last execution.

For **End Date**, use the format YYYY/MM/DD. For the **End Time**, use the format HH:MM:SS (using a 24-hour clock).

For example, if you want the service executions to stop on June 4, 2010 at 2:00:00 a.m., specify 2010/06/04 for **End Date** and 02:00:00 for **End Time**. Omitting **End Date** indicates that you want this service to execute for an indefinite period of time. If you omit **End Time**, the server uses the current time.

- **Run Mask**. Specific months, dates (1-31), days of the week (Sunday-Monday), hours, minutes, and seconds when you want a task to run. You can select one or more items from each category. To select multiple items, press the Ctrl key while making your selections. If you do not select any items from a category, Integration Server assumes all items for the selection.

- **Repeat after completion**. Whether to wait for the previous execution of a service to complete before starting the next.

For example, suppose the GetData service is scheduled to run every minute, but sometimes takes longer than that to complete. By default, Integration Server will start the next execution even though the previous one has not yet completed.

Select**To**

If you check the **Repeat after completion** box, the server will wait for the service to complete before running the next execution of the service. Executions that could not run while the service was running are delayed.

For more information about using the Complex Repeating options see [“Complex Repeating Option” on page 727](#).

Note: Integration Server does not skip or repeat the scheduled tasks because of transitions to and from Daylight Saving Time. For more information about how Integration Server handles transitioning to and from Daylight Saving Time, see [“How Transitioning to or from Daylight Savings Time Affects Scheduled Tasks” on page 732](#).

7. Click **Save Tasks**.

Viewing Scheduled User Tasks

Note:

If your Integration Server runs as part of a group of servers connected to the same database, all tasks will be visible from the **Server > Scheduler** screens of all Integration Servers connected to the database.

➤ To view scheduled user tasks

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.

The Server Scheduler screen displays the list of scheduled user tasks. Integration Server automatically deletes the expired scheduled user tasks from the Server Scheduler screen. If you do not want Integration Server to delete the expired tasks until the next server restart, set the `watt.server.scheduler.deleteCompletedTasks` parameter to false. For more information about this parameter, see [“Server Configuration Parameters” on page 1017](#).

Filtering the List of Scheduled Tasks

By default, the **Server > Scheduler** screen lists all the user-defined tasks that are scheduled to run. You can use the filter to limit the tasks to be displayed, making the list shorter and more manageable.

➤ To filter the list of scheduled tasks

1. In the **Server** menu of the Navigation panel, click **Scheduler**. The list of user-defined tasks will be displayed.

- Click **Filter Services**. The filtering options will appear above the list of user tasks.

Note:

When **Filter Services** is enabled, any changes to the Integration Server (such as new tasks, etc.) will not be reflected in the list of scheduled tasks. When you click **Show All Services** and return to normal mode, the list will be updated.

- Select some or all of the following options:

Option	Description
Service Name	<p>The string you want to submit to the filter. Filter criteria can be literals or a combination of literal and wild-card characters. The "*" (asterisk) and "?" (question mark) are the only supported wild-card characters. Leaving the filter criteria blank includes all services.</p> <p>The service names in the Service Name field are case-sensitive. For example, if you enter "Wm*", the filter will ignore any services beginning with "wm".</p>
Status Is Active	Controls whether only active services are displayed. If not checked, services with all statuses are displayed.
Hide Remote Tasks	Controls whether only tasks that run on the current server are displayed. This also includes tasks scheduled to run on the server when the Target Node is set to run on All servers in the cluster or Any server connected to the database. Tasks scheduled to run only on other servers are not displayed.

Updating Scheduled User Tasks

Note:

If your Integration Server is part of a cluster and you are updating the characteristics of a task that has been scheduled to run on all Integration Servers in the cluster, you must make the changes to the parent task. For more information about scheduled tasks that run on all servers in a cluster, see ["Target Node Options" on page 730](#).

> To update a scheduled user task

- Open the Integration Server Administrator if it is not already open.
- In the **Server** menu of the Navigation panel, click **Scheduler**.
- Click the service name for the user task you want to update.

4. Update the scheduling options for the selected user task. For information about the options you can specify, see [“Scheduling a User Task” on page 716](#).
5. Click **Update Tasks**.

Suspending User Tasks

You can suspend a single user task or all the scheduled user tasks in an Integration Server.

Suspending a Single User Task

When you suspend a user task, it remains scheduled, but does not execute until you resume its execution. If a task expires while suspended, the server marks it *Expired*.

Note:

If your Integration Server is part of a cluster and you are suspending a task that has been scheduled to run on all Integration Servers in the cluster, you can suspend the child tasks individually or you can suspend all the tasks at once by suspending the parent task. For more information about scheduled tasks that run on all servers in a cluster, see [“Target Node Options” on page 730](#).

➤ To suspend a scheduled user task

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.
3. Locate the task in the **Services** list, and click the ✓ icon in the **Status** column to suspend the task. The server displays a screen to confirm you want to suspend the task. Click **OK**. The server replaces the ✓ icon with **Suspended** to indicate that the task is now suspended.

Suspending All User Tasks

Suspending the tasks affects only the scheduled user tasks and not system tasks.

When you suspend all the scheduled user tasks in an Integration Server:

- Integration Server will not initiate any scheduled user tasks. However, you can schedule new tasks and perform other task-specific actions.
- The scheduled user tasks that are currently running will continue to execute.
- The suspend action will not affect the system tasks and they will continue to run.

➤ To suspend all scheduled user tasks

1. Open Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.
3. Click **Pause Scheduler**. Integration Server displays a message prompting you to confirm that you want to suspend the tasks. Click **OK**.

Integration Server will initiate the scheduled user tasks only after you have resumed the tasks.

Resuming Suspended User Tasks

You can resume all scheduled executions of a user task that you have suspended. You can also resume the execution of all the scheduled user tasks in an Integration Server.

Resuming a Suspended User Task

Perform the following procedure to resume all scheduled executions of a task that has been suspended.

Note:

If your Integration Server is part of a cluster and you are resuming a task that has been scheduled to run on all Integration Servers in the cluster, you can resume the child tasks individually or you can resume all the tasks at once by resuming the parent task. For more information about scheduled tasks that run on all servers in a cluster, see [“Target Node Options” on page 730](#).

To resume execution of a suspended user task

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.
3. Locate the task in the **Services** list, and click **Suspended** in the Active column to activate the task. The server displays a screen to confirm you want to resume the task. Click **OK**.

The server replaces **Suspended** with the **Active** ✓ icon to indicate that the task is available to execute again.

Resuming All Suspended User Tasks

When you resume the execution of all the scheduled user tasks in an Integration Server:

- Integration Server will start initiating the scheduled user tasks again.
- If any of the scheduled user tasks that were running when the tasks were suspended are still running, they will continue to execute.
- Individual tasks that you have suspended by clicking the ✓ icon in the **Status** column will remain suspended until you specifically activate it.

➤ To resume all scheduled user tasks

1. Open Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.
3. Click **Resume Scheduler**. Integration Server displays a message prompting you to confirm that you want to resume the initiation of tasks. Click **OK**.

Integration Server will proceed to initiate the scheduled user tasks.

Important:

If a task misses its scheduled execution time because the execution of tasks was suspended at that time, the next execution of the task depends on the value of the **If the Task Is Overdue** setting. Based on this setting, Integration Server will start the task as soon as you resume the initiation of tasks, skip this execution of the task, or suspend the task and wait for administrator action. For more information, see [“Scheduling a User Task” on page 716](#).

Canceling a Scheduled User Task

When you cancel a scheduled task, the server permanently removes it from the database that holds the job queue.

Note:

If your server is part of a cluster and you are canceling a task that has been scheduled to run on all servers in the cluster, you can cancel the child tasks individually or you can cancel all the tasks at once by canceling the parent task. For more information about scheduled tasks that run on all servers in a cluster, see [“Target Node Options” on page 730](#).

➤ To cancel a scheduled user task

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.
3. Click the **✗** icon in the **Remove** column for the user task you want to cancel. The server issues a prompt to verify that you want to cancel the user task. Click **OK**.

Viewing the Scheduled System Tasks

Integration Server needs to perform system tasks periodically, such as expiring sessions. Integration Server schedules these tasks automatically.

➤ To view the scheduled system tasks

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Scheduler**.
3. Click **View system tasks**.

The server displays the **View System Tasks** screen. It lists the names of each scheduled task, the next date and time the server is to execute the task, and how often (**Interval**) the server is to execute the task.

Note:

The **View System Tasks** screen shows the tasks for local server only; if you are running a group of servers connected to the same database, you will not see the system tasks for other servers connected to the database.

Simple Repeating Option

When you use the simple repeating option, the service repeats based on a time interval you specify.

Setting	Indicates
Start Date	The date on which you want the server to execute the service for the first time. Use the format YYYY/MM/DD to specify the date. If you leave this field blank, the server starts the task on the current day
Start Time	The time at which you want the server to begin executing the service. Use the format HH:MM:SS to specify the time (using a 24-hour clock). If you leave this field blank, the server starts the task immediately.
End Date	The date on which you want the server to execute the service for the last time. Use the format YYYY/MM/DD to specify the date. If you leave this field blank, the server executes the service for an indefinite period of time or until you cancel the scheduled user task.
End Time	The time on the last date at which you want the server to execute the service. Use the format HH:MM:SS to specify the time (using a 24-hour clock). If you leave this field blank, the server uses the current time.
Repeating/ Repeat after Completion	<p>Whether Integration Server should wait for the previous execution of a service to complete before starting the next.</p> <p>For example, suppose the GetData service is scheduled to run every minute, but sometimes takes longer than that to complete. By default, the server will start the next execution even though the previous one has not yet completed. If you check the Repeat after completion box, Integration Server will wait for the service to complete before running the next execution of the service. Executions that could not run while the service was executing are delayed.</p>

Setting	Indicates
Interval	Time interval, in seconds, at which you want the service to execute. For example, if you want the server to execute the service every 24 hours, specify 86400 seconds for the interval.

The following example shows how to use the Simple Repeating option settings:

Service to execute	For this setting	Specify
Every hour on July 1st in the year 2010.	Start Date	2010/07/01
	Start Time	00:00:00
	End Date	2010/07/01
	End Time	00:00:00
	Interval	3600

Complex Repeating Option

With the **Complex Repeating** option, the service repeats based on complex intervals that you specify. This option offers the greatest flexibility for specifying when you want a service to execute.

Specify any combination of the following settings to indicate when and how often you want the service to execute:

Setting	Indicates
Start Date	The date on which you want the server to execute the service for the first time. Use the format YYYY/MM/DD to specify the date. If you leave this field blank, the server executes the task at the first date specified by the remaining settings.
Start Time	The time at which you want the server to begin executing the service. Use the format HH:MM:SS to specify the time (using a 24-hour clock). If you leave this field blank, the server uses the current time.
End Date	The date on which you want the server to execute the service for the last time. Use the format YYYY/MM/DD to specify the date. If you leave this field blank, the server executes the service for an indefinite period of time or until you cancel the scheduled user task.
End Time	The time on the last date at which you want the server to execute the service. Use the format HH:MM:SS to specify the time (using a 24-hour clock). If you leave this field blank, the server uses the current time.
Repeating/ Repeat after Completion	Whether Integration Server should wait for the previous execution of a service to complete before starting the next.

Setting	Indicates
	<p>Note:Integration Server does not count half-finished service executions as completed tasks. If a task fails before completing, Integration Server considers that task as overdue. The next execution of the task depends on whatever action is set for the If the Task is Overdue option. For more information, see “Scheduling a User Task” on page 716</p> <p>If you want Integration Server to wait for a service to complete execution before it starts the next scheduled execution of the service, check this box.</p> <p>For example, suppose the GetData service is scheduled to run every minute, but sometimes takes longer than that to complete. By default, the Integration Server will start the next execution even though the previous one has not yet completed. If you check the Repeat after completion box, Integration Server will wait for the service to complete before running the next execution of the service. Executions that could not run while the service was executing are delayed.</p>
Run Mask	<p>Specific months, days (1-31), days (Sunday-Saturday), hours, and minutes you want the service to run.</p> <p>You can select one or more items from each category. To select multiple items, press the Ctrl key while making your selections. If you do not select any items from a category, Integration Server assumes all items for the selection. For example, if you do not specify a month, Integration Server assumes you want the service to execute every month. If you do not select any items for any of the settings, the Integration Server assumes you want the service to execute every month, every day, all week days, every hour, and every minute; in other words, the service runs every minute from the time you add the task.</p>

Integration Server combines all your selections to determine when to execute the service.

The following shows examples of how to use the **Complex** option settings:

Service to execute	For this setting	Specify
The 28th day of every month at midnight for the year 2010.	Start Date	2010/01/01
	Start Time	00:00:00
	End Date	2010/12/31
	End Time	00:00:00
	Months	no selection
	Month Days	28

Service to execute	For this setting	Specify
	Week Days	no selection
	Hours	0
	Minutes	0
Every Monday in the months of January, February, and March at 2:30 p.m. for an indefinite period of time.	Start Date	leave blank
	Start Time	leave blank
	End Date	leave blank
	End Time	leave blank
	Months	January, February, March
	Month Days	no selection
	Week Days	Monday
	Hours	14
	Minutes	30
	Every hour of every Tuesday of the month of June, 2010.	Start Date
Start Time		00:00:00 (or leave blank)
End Date		2010/06/30
End Time		00:00:00 (or leave blank)
Months		June
Month Days		no selection
Week Days		Tuesday
Hours		no selection
Minutes		0
Every minute of every hour of every Tuesday of the month of June, 2010.	Start Date	2010/06/01
	Start Time	00:00:00 (or leave blank)
	End Date	2010/06/30
	End Time	00:00:00 (or leave blank)

Service to execute	For this setting	Specify
	Months	June
	Month Days	no selection
	Week Days	Tuesday
	Hours	no selection
	Minutes	no selection

Target Node Options

If you are running a group of servers connected to the same database (clustered or non-clustered), you can control on which server a task runs. You can specify that a task run on the current server, another specific server, all servers in a cluster, or any servers connected to the database:

Setting	Indicates
<specific_server>	The task runs only on the server you specify.
Any server	<p>The task runs on any server connected to the database. Use this option if the task only needs to run on one server and it doesn't matter which one. For example, in a clustered environment, if all servers in the cluster share a single database for a parts inventory application, and a particular function needs to run on that database once a day, any server in the cluster can perform that function. The Any server option is the default setting when clustering is enabled.</p> <p>Note: The Any server option does not specify an order in which servers are used to execute tasks. In other words, no load balancing is performed. Instead, an instance of the scheduler runs on each server connected to the database. Periodically, each instance checks the database in which information about scheduled jobs is stored. The first scheduler instance to find a task that is due to start runs it, then marks the task as complete in the database where information about scheduled jobs is stored. The scheduler instances running on the other servers connected to the database then know not to run the task. This behavior will not change if you install a third-party load balancer.</p>
All servers	For clustered Integration Servers only. If you select All servers , the task runs on all servers in the cluster. For example, suppose you run an application on each server in the cluster, and each server maintains its own database for that application. If you need to run a cleanup task against all the databases every day, then from one server you can schedule a task to run every day on all the servers in the cluster.

Setting	Indicates
	When you schedule a task to run on all servers in the cluster, the server divides the task into a main or <i>parent</i> task, and a <i>child</i> task for each server in the cluster. See “ Tasks in a Clustered Environment ” on page 731 below for more information about these tasks.

Tasks in a Clustered Environment

When you schedule a task to run on all servers in the cluster, the server divides the task into a main or *parent* task, and a *child* task for each server in the cluster.

You might see different statuses among the parent and child tasks. For example, you might have a situation where the parent status is Active, one child status is Active, and the other child status is Suspended. In general, the status of the parent task will be Active if at least one child task is active or running, Suspended if all child tasks are suspended, or Expired, if all child tasks are expired.

Although you can perform some actions (activate, suspend) individually on the child tasks, if you want to change the characteristics of a task, you must do so through the parent task.

To keep the parent and child tasks in synch, Integration Server automatically updates child tasks when you:

- Change the parent task. For example, if you change the end date of the parent task, Integration Server will automatically change the end date in the associated child tasks.
- Add or delete a cluster node. For example, if you add another Integration Server to your cluster, Integration Server automatically creates a child task for that node at Integration Server startup.
- Delete a child task. For example, if you delete a child task for a cluster node that still exists, Integration Server will automatically recreate the child task at Integration Server startup, or when you update the parent task.

The following picture shows how Integration Server Administrator displays parent and child tasks on the **Server > Scheduler** screen.

Tasks in a Clustered Environment

If you schedule a task to run on *all* servers in the cluster, the server divides the task into Parent and Child tasks.

If you schedule a task to run on *any* server in the cluster, the server shows the target server as *Any Server*.

One-Time and Simple Interval Tasks										
ID	Service	Description	Queue Name	Last Error	Run As User	Target	Interval (sec)	Next Run (sec)	Status	Remove
c7899460-e0ab-11e5-91ba-ee8977bd8c7b	wm.server.ping	Any Server	N/A	N/A	Developer	Any Server	24000.0	23937.5	Active	✗
d9553e60-e0ab-11e5-a557-cab100280c23	wm.server.ping	Local Task	N/A	N/A	Developer	ICWin2k8R264BVT02	24000.0	23967.4	Active	✗
ec5f34c0-e0ab-11e5-9d6a-c00be2d86b78	wm.server.ping	All Servers	N/A	N/A	Developer	All Servers	24000.0	N/A	Active	✗
ec722080-e0ab-11e5-b855-c00be2d86b78	wm.server.ping	All Servers	N/A	N/A	Developer	ICWin2k8R264BVT02	24000.0	23999.3	Active	✗

Parent Task (row 3): ec5f34c0-e0ab-11e5-9d6a-c00be2d86b78

Child Task (row 4): ec722080-e0ab-11e5-b855-c00be2d86b78

In a Child task, you cannot link to the service.

In the Parent task, the target server is shown as *All Servers*.

If you suspend, resume, or cancel a parent task, the server suspends, resumes, or cancels the associated child tasks as well.

Note:

When using the Scheduler in a cluster of Integration Servers, you can ensure that repeating and complex tasks run at the desired times by synchronizing the system clocks of all the servers.

How Transitioning to or from Daylight Savings Time Affects Scheduled Tasks

Transitions to and from Daylight Savings Time (DST) affect scheduled tasks in Integration Server in the following ways:

- When transitioning to DST, the system clock time is advanced by one hour. However, Integration Server will not skip the scheduled tasks that were scheduled to be run during the hour that is skipped because of transitioning to DST. If a task was scheduled to run in the hour that was skipped, Integration Server will run the task in the ensuing hour. For example, if you have scheduled a task to run at 1:30:00 a.m. and if system clock time transitions to DST at 1:00:00 a.m., Integration Server will process the task at 2:30:00 a.m. system clock time.
- When transitioning from DST, the system clock time is set back one hour. For a repeating task scheduled to run in the DST overlap, Integration Server does one of the following:
 - If a task includes an hours mask (e.g., "1:15" for running every date at 0115), then the task will run only once, at the pre-overlap ("first") matching time. For example, if a task is executed at 1:15:00 a.m. and if the system clock time transitions to DST at 2:00:00 a.m., which sets the clock back to 1:00:00 am, Integration Server does not process the task again at 1:15:00 a.m.
 - If a task does not have an hours mask (e.g., "hh:15", meaning run at 15 minutes past each hour), then the task will run at the pre-overlap time (the "first" 1:15), and also at the overlap time (the "second" 1:15).
- The simple and complex repeating tasks are unaffected by the transitions to and from DST. Integration Server processes the repeating tasks at the specified intervals. That is, if a task is scheduled to run every hour, Integration Server will execute the task at hourly intervals regardless of the system clock time.

38 Caching Service Results

■ What Is Caching?	734
■ When Are Cached Results Returned?	734
■ Using a Public Cache for Service Results Caching	735
■ Resetting the Cache	736
■ Monitoring Service Cache Usage	737
■ Viewing Service Results in a Public Cache	737

What Is Caching?

Caching is an optimization feature that can improve the performance of services.

You indicate the services for which you want to use caching from Software AG Designer. When you enable caching for a service, Integration Server saves the entire contents of the pipeline after invoking the service in a local or distributed cache for the period of time that you specify. The pipeline includes the output fields explicitly defined in the cached service, as well as any output fields produced by earlier services in the flow. When Integration Server receives subsequent requests for a service with the *same set of input values*, Integration Server returns the cached result to the client instead of invoking the service again.

Caching can significantly improve response time of services. For example, services that retrieve information from busy data sources such as high-traffic commercial web servers could benefit from caching. Integration Server can cache the results for all types of services: flows, Java services, and C/C++ services.

The goal for caching is to strike the right balance between data concurrency and memory usage. To gauge the effectiveness of your cache, you can monitor its performance by viewing service statistics from the Integration Server Administrator and adjust your caching values accordingly.

You set the controls for caching a service from Designer. For instructions, see *webMethods Service Development Help*.

When Are Cached Results Returned?

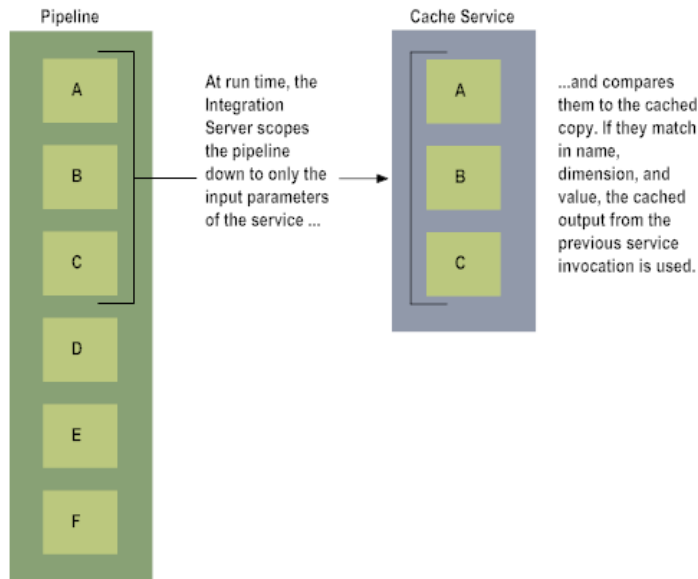
When you enable caching for a service in Software AG Designer, Integration Server handles the cached results differently, depending on whether the service has input parameters. It is recommended that a cached service has input parameters.

When a cached service has input parameters, at run time Integration Server scopes the pipeline down to only the declared input parameters of the service. Integration Server compares the scoped-down inputs to the previously stored copy of the inputs. If a cached entry exists with input parameters that have the same values, Integration Server returns the cached results from the previous service invocation.

Note:

If a cached entry with input parameter values that are identical to the current invocation does not exist in the cache, Integration Server executes the service and stores the results in the cache.

Pipeline Inputs Are Compared to the Cached Copy at Run Time



When a cached service does not have input parameters (for example, a date/time service) and previous results do not exist in the cache, at run time Integration Server executes the service and stores the results. When the service executes again, Integration Server uses the cached copy. In other words, Integration Server does not use the run-time pipeline for the current service invocation; you will always receive cached results until the cache expires.

Important:

If a cached service input signature includes a Document Reference or Document Reference List variable and the referenced document type changes or is modified, you must reset the service cache. If you do not reset it, Integration Server uses the old cached input parameters at run time until such time as the cached results expire. You can reset the cache from Designer or Integration Server Administrator. For more information about resetting service cache from Integration Server Administrator, see [“Resetting the Cache” on page 736](#).

Using a Public Cache for Service Results Caching

By default, Integration Server stores cached service results in the local ServiceResults system cache, which is part of the SoftwareAG.IS.Services system cache manager. If you want multiple Integration Server to be able to access cached service results, you can store cached service results in a public cache, including a distributed cache, instead. Only Integration Servers that have the same packages installed on them can share a distributed cache.

Keep the following information in mind when using a public cache for service results caching:

- The cache must be configured so that elements do not expire once they are placed in the cache. To accomplish this, select the **Eternal** check box for the cache (see [“Creating a Cache” on page 783](#)).
- The duration of cached service results depends on the Cache expire property value for the service and the `watt.server.cache.flushMins` server configuration parameter value.

- By default, when Integration Server returns cached results for a service, it returns the actual value of the cached results. If you want Integration Server to return a reference instead, clear the **Copy on Read** and **Copy on Write** check boxes for the cache (see “[Creating a Cache](#)” on page 783).
- When a public cache used for service results caching is disabled or a public cache manager that contains the cache used for service results caching is shut down, Integration Server does not cache or retrieve service results. Instead, Integration Server executes the service.
- When a public cache used for service results caching is enabled or the public cache manager that contains the public cache used for service results is started, Integration Server re-initializes the cache.

For detailed information about creating a public cache, public cache manager, and a distributed cache, see “[Configuring Ehcache on Integration Server](#)” on page 749.

Note:

In Integration Server Administrator, the **Server > Service Usage** page displays statistics about service results. For a distributed cache, these statistics are for the current Integration Server instance only. Integration Server Administrator does not provide aggregated statistics for all the Integration Servers using the same distributed cache for service results caching.

To configure Integration Server to use a public cache for service results caching, you specify the name of the cache and cache manager in the server configuration parameters `watt.server.serviceResults.cache` and `watt.server.serviceResults.cacheManager`, respectively.

Resetting the Cache

You can reset the cache for all services or you can reset the cache for a specific service.

Note: Integration Server resets the cache for a service automatically whenever any edits are made to the service. However, if the input signature includes a document reference variable and the referenced document type changes, you must reset the service cache. If you do not reset it, Integration Server uses the old cached input parameters at run time until such time as the cached results expire.

Resetting the Cache for All Services

When Integration Server resets the cache for all services, it removes all cached service results for all services from memory.

You can also use the `pub.cache.serviceResults:resetServerCache` service to reset the cache for all services.

➤ To reset the cache for all services

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Service Usage**.

3. Click **Reset Server Cache** to reset the caches of all the listed services.

Resetting the Cache for a Specific Service

When Integration Server resets the cache for a service, it removes the cached service results for that service from memory.

You can also use the `pub.cache.serviceResults:resetServiceCache` service to reset the cache for a specific service.

> To reset the cache for a specific service

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Service Usage**.
3. Select the name of the service for which you want to reset the cache. An information screen for that service appears.
4. Click **Reset Service Cache**.

Monitoring Service Cache Usage

Use the following procedure to monitor the performance of the service cache.

> To monitor the performance of your cache

1. Open the Integration Server Administrator if it is not already open.
2. In the **Server** menu of the Navigation panel, click **Service Usage**.

The **Service Usage** screen displays the current results of cache control settings for each cache-controlled service. By default Integration Server Administrator displays the services in alphabetical order.

Tip:

Select the **Show running services on top** check box to display all the currently running services in the Integration Server together at the top of the screen. Currently running services are identified by a number in brackets at the right of the service name. The number identifies how many instances of the service, if any, are currently running.

Viewing Service Results in a Public Cache

Use the following procedure to view the cached service results for a particular service. You can view cached service results stored in a public cache but not in the ServiceResults system cache.

You can also use the `pub.cache.serviceResults:listServiceCache` service to list the cached service results for a particular service. Before you can use this procedure or the service, make the service results cache searchable and set it to allow automatic indexing for keys. To accomplish this, select the **Searchable** check box and the **Key** check box next to **Allow Automatic Indexing** (see “[Creating a Cache](#)” on page 783).

Note:

Exposing the contents of the service results cache may represent a security risk.

> To view a service results cache for a service

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation Panel, click **Management**.
3. In the **Packages** list, click the package that contains the service for which to view cached service results.
4. Click **Browse Services in *packageName***
5. In the **Services** list, click the service for which to view cached service results.
6. On the **Packages > Management > *packageName* > Services > *serviceName*** page, click **List Service Cache**.

Integration Server displays a list of the cached results for the service. For each cached element, Integration Server displays the key for the cached element, the cached service inputs, and the associated cached service outputs.

39 Configuring Guaranteed Delivery

■ About Guaranteed Delivery	740
■ Configuring the Server for Guaranteed Delivery	741
■ Administering Guaranteed Delivery	745

About Guaranteed Delivery

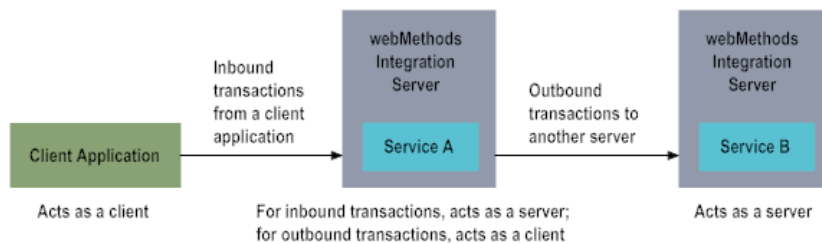
Use the guaranteed delivery capabilities of Integration Server to ensure guaranteed one-time execution of services.

Integration Server guaranteed delivery capabilities ensure that the following occur despite transient failures:

- Requests to execute services from clients are delivered to the server
- Services are executed once, and only once
- Responses from the execution of the services are delivered to the client

The webMethods guaranteed delivery capabilities protect against transient failures that might occur on the network, in the client, or on the server. A transient failure is a failure that can correct itself during a specified period of time. If a request cannot be delivered to the server due to a transient failure, the request is resubmitted; if the problem has corrected itself, the request is successfully delivered on a subsequent attempt. You determine what constitutes a transient error by specifying a time-to-live (TTL) period for a guaranteed delivery transaction and, optionally, the number of times a transaction should be retried.

Because an Integration Server can act as either a server or a client in a guaranteed delivery transaction, the guaranteed delivery capabilities of the server handle both inbound transactions and outbound transactions. When a client invokes a service on a server, the server is acting as a server. If a service uses guaranteed delivery to invoke a service on another Integration Server, the server that invokes the service is the client.



Note:

The Integration Servers that participate in a guaranteed-delivery transaction must both be running the same version of Integration Server software.

The guaranteed delivery capabilities allow you to build robust, transaction-based client applications without having to embed complex error handling code to respond to transient failures.

Important:

Use the guaranteed delivery capabilities with stateless (i.e., atomic) transactions because state information cannot be maintained from one request to the next. As a result, guaranteed delivery capabilities cannot be used with multi-request conversational services.

For more information about guaranteed delivery, see the *Guaranteed Delivery Developer's Guide*.

Configuring the Server for Guaranteed Delivery

Integration Server uses various configuration settings for guaranteed delivery transactions. Most of the settings have defaults. In general, you will want to use the defaults. You can change these setting by using the **Settings > Extended** screen of the Integration Server Administrator as described in [“Working with Extended Configuration Settings” on page 127](#).

For guaranteed delivery to work, the ISInternal functional alias (specified on the **Settings > JDBC Pools** screen) must be configured to point to either the embedded IS Internal database or to an external RDBMS. For information about connecting Integration Server to database components, see [“Connecting Integration Server to Database Components in an External RDBMS” on page 139](#).

For information about using guaranteed delivery with server clustering, see the *webMethods Integration Server Clustering Guide* .

Settings Shared by Both Inbound and Outbound Transactions

The following settings apply to both inbound and outbound guaranteed delivery transactions.

watt.server.txMail

Use the watt.server.txMail setting to specify the e-mail address of an administrator to notify when guaranteed delivery capabilities are disabled due to error (for example, if the server encounters a disk full condition). An example of using this setting is watt.server.txMail=ISAdmin@YourCompany.com.

There is no default for this setting.

watt.server.smtpServer

Use the watt.server.smtpServer setting to specify the domain name (e.g., purple.webmethods.com) or IP address (e.g. 132.196.19.22 or 2001:db8:85a3:8d3:1319:8a2e:370:7348) of the SMTP server you want the Integration Server to use when sending an e-mail message about an error during guaranteed delivery. An example of using this setting is watt.server.smtpServer=132.196.19.22 or watt.server.smtpServer=2001:db8:85a3:8d3:1319:8a2e:370:7348

There is no default for this setting.

When an administrator receives an e-mail notification of an error, the administrator should correct the problem, then use Integration Server Administrator to re-initialize guaranteed delivery capabilities. For instructions on how to re-initialize guaranteed delivery, refer to [“Reinitializing Guaranteed Delivery” on page 745](#).

watt.tx.vm.id

Use the watt.tx.vm.id setting to specify an Integration Server ID. Use this parameter when multiple Integration Servers are running on the same host machine and they all use guaranteed delivery. By specifying a unique ID for each Integration Server, you prevent the creation of duplicate guaranteed delivery transaction IDs. The value must be an integer from 0 through 2147483647.

The default is 0.

Settings for Inbound Transactions

For inbound transactions, the server maintains a *job store* of transactions and the status of each. Periodically, the server *sweeps* the job store to remove expired transactions; that is, to remove transactions that have an elapsed time-to-live (TTL) period. For inbound requests, the client must specify the TTL for a transaction.

In addition to the job store, the server maintains an audit-trail log of all operations it performs for inbound transactions.

The following table identifies the inbound transaction settings that you can configure.

You can configure:	Using this setting
How often the server sweeps the job store to remove expired transactions	<code>watt.server.tx.sweepTime</code>
How the server updates the status of PENDING transactions when a heuristic failure occurs	<code>watt.server.tx.heuristicFailRetry</code>

watt.server.tx.sweepTime

Use the `watt.server.tx.sweepTime` setting to specify the number of seconds between sweeps (clean up) of the job store of inbound transactions. The server sweeps the job store to remove expired transactions.

The default is: 60 seconds

watt.server.tx.heuristicFailRetry

Use the `watt.server.tx.heuristicFailRetry` setting to indicate whether the server is to re-execute services for transactions in the job store that are PENDING when the server is restarted after a failure. If a transaction is PENDING, the service began but did not complete execution when the server failed.

Because the server cannot determine the exact status of a service request, the server considers the guaranteed transaction to have encountered a heuristic failure. You can configure the server to respond to heuristic failures as appropriate. The default `watt.tx.heuristicFailRetry` setting causes the server to execute a service at least one time at the risk of re-executing it a subsequent time after a heuristic failure. Alternatively, you can reconfigure the setting to guarantee that a service is executed at most one time at the risk of not executing a service due to a heuristic failure.

If the `watt.tx.heuristicFailRetry` setting is `true`, the server resets the transaction status from PENDING to NEW, and the server will retry the service. When the setting is `true`, a request to execute a service can only fail if the transaction expires before the server executes the service. (The client specifies the settings that indicate when a transaction expires.)

If the `watt.tx.heuristicFailRetry` setting is `false`, the server resets the transaction status from PENDING to FAIL to indicate the heuristic failure; the server does not retry the service. When the setting is `false`, a request to execute a service can fail due to a heuristic failure or due to the transaction expiring.

The default is: `true`

Settings for Outbound Transactions

You can disable the use of guaranteed delivery for outbound transactions. However, if you allow guaranteed delivery for outbound transactions, the server maintains a separate job store for the transactions. Similar to the inbound job store, the server keeps the status of each transaction in the outbound job store. If a service request fails, the server waits a specified amount of time before resubmitting the request. The server periodically processes the job store to identify transactions that it needs to submit.

The server maintains a thread pool to service pending outbound requests. You can configure how many client threads the server should maintain in the thread pool.

The server also maintains a separate audit-trail log of all operations it performs for outbound transactions.

The following table identifies the outbound transaction settings that you can configure.

You can configure:	Using this setting
Whether you want to disable guaranteed delivery for outbound transactions.	<code>watt.tx.disabled</code>
The default TTL value for outbound transactions.	<code>watt.tx.defaultTTLmins</code>
How long the server should wait before resubmitting failed requests.	<code>watt.tx.retryBackoff</code>
How often the server processes the job store to identify transactions that it needs to submit.	<code>watt.tx.sweepTime</code>
How many client threads the server should maintain in the thread pool that it uses to service pending requests.	<code>watt.tx.jobThreads</code>

watt.tx.disabled

Use the `watt.tx.disabled` setting to specify that you want to disable the use of guaranteed delivery for outbound requests. By default, the server allows the use of guaranteed delivery for outbound transactions. The default is `false`. If an unexpected exceptional conditional is encountered, guaranteed delivery may be disabled by the server. In this case, the `watt.tx.disabled` property will be set to `true`.

The default is: `false`.

watt.tx.defaultTTLmins

Use the `watt.tx.defaultTTLmins` setting to specify the default time-to-live (TTL) value for outbound guaranteed delivery transactions. Specify the number of minutes you want the server to maintain outbound transactions in the job store when a service initiating an outbound transaction does not specify a TTL value.

The default is: 30.

watt.tx.retryBackoff

Use the `watt.tx.retryBackoff` setting to specify the number of seconds to wait after a service request failure before the Job Manager resubmits the request to execute the service to the Integration Server.

The default is: 60.

watt.tx.sweepTime

Use the `watt.tx.sweepTime` setting to specify the number of seconds between sweeps of the job store of outbound transactions. The server sweeps the job store to identify transactions that it needs to submit.

The default is: 60.

watt.tx.jobThreads

Use the `watt.tx.jobThreads` setting to specify the number of client threads you want to make available in a thread pool to service pending requests.

The default is: 5.

Specifying an E-Mail Address and SMTP Server for Error Messages

When you configure guaranteed delivery, you must specify the e-mail address to which Integration Server can issue an error message if guaranteed delivery becomes disabled. In addition, you must specify the domain name or IP address of the SMTP server you want to handle these e-mail messages. For instructions on specifying the e-mail address and SMTP server, refer to [“Sending Messages About Critical Issues to E-mail Addresses”](#) on page 220.

Using Guaranteed Delivery with Multiple Servers that Share an ISInternal Database

If you use multiple Integration Servers that are not in an Integration Server cluster, but the servers are configured with their `ISInternal` functional alias pointing to the same database, you need to configure each Integration Server correctly process guaranteed delivery jobs.

For each server sharing the `ISInternal` database, set the `watt.server.db.share.ISInternal` server property to `true`, which lets the server know that it is sharing the database and that it needs to coordinate with other servers.

Additionally, although the servers are not in an Integration Server cluster, they are sharing the Guaranteed Delivery tables. As a result, you need to configure the following clustering server properties identically for all servers:

- `watt.server.tx.cluster.lockTimeoutMillis`
- `watt.server.tx.cluster.lockBreakSecs`
- `watt.server.tx.cluster.jobPendingWait`

To view or change server property settings, use the **Settings > Extended** screen from the Integration Server Administrator as described in “[Working with Extended Configuration Settings](#)” on page 127. For information about server properties, see the “[Server Configuration Parameters](#)” on page 1017.

Important:

After changing these server properties, you must restart Integration Server for the changes to take effect.

Administering Guaranteed Delivery

When you initialize the server, it initializes guaranteed delivery capabilities. You can use the Integration Server Administrator to shut down, re-initialize, and test guaranteed delivery.

Shutting Down Guaranteed Delivery

You can shut down and re-enable guaranteed delivery capabilities without having to shut down the server.

You might want to shut down guaranteed delivery to perform some administration functions, such as correcting configuration errors or starting a new audit-trail log. (To start a new audit-trail log, move or rename the existing log; the server automatically starts a new log if one does not already exist.)

➤ To shut down guaranteed delivery

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. In the list of packages, click **WmPublic**.
4. Click **Browse Services in WmPublic**.
5. In the list of services, click **pub.tx:shutdown**.
6. Click **Test shutdown**. The server displays the test screen for the service.
7. Click **Test (without inputs)**. The server disables the guaranteed delivery capabilities for inbound transactions.

Reinitializing Guaranteed Delivery

Reinitialize guaranteed delivery if it becomes disabled. The procedures for reinitializing guaranteed delivery are different for inbound transactions and outbound transactions.

Reinitializing Guaranteed Delivery for Inbound Transactions

If you shut down the guaranteed delivery capabilities to correct a configuration problem or to make an administrative change, you can re-initialize guaranteed delivery using the Integration Server Administrator.

You can also use this procedure to reinitialize guaranteed delivery if it becomes disabled due to an error (for example, because of a disk full condition or if the job store database becomes inaccessible). Reinitialize guaranteed delivery after you correct the problem.

➤ To reinitialize guaranteed delivery for inbound transactions

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. In the list of packages, click **WmPublic**.
4. Click **Browse Services in WmPublic**.
5. In the list of services, click **pub.tx:init**.
6. Click **Test init**. The server displays the test screen for the service.
7. Click **Test (without inputs)**. The server reinitializes the guaranteed delivery capabilities for inbound transactions.

Reinitializing Guaranteed Delivery for Outbound Transactions

If guaranteed delivery capabilities for outbound transactions become disabled due to an error (for example, because of a disk full condition or if the job store database becomes inaccessible), use this procedure to reinitialize guaranteed delivery after you correct the problem.

➤ To reinitialize guaranteed delivery for outbound transactions

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. In the list of packages, click **WmPublic**.
4. Click **Browse Services in WmPublic**.
5. In the list of services, click **pub.tx:resetOutbound**.

6. Click **Test resetOutbound**. The server displays the test screen for the service.
7. Click **Test (without inputs)**. The server reinitializes the guaranteed delivery capabilities for outbound transactions.

40 Configuring Ehcache on Integration Server

■ What is Ehcache?	750
■ Caching Configurations	750
■ Understanding Caches and Cache Managers	753
■ Cache Manager Configuration Files	755
■ Installing, Viewing, and Changing the Terracotta License	756
■ Configuring an On-Heap Cache	758
■ Configuring a BigMemory Cache	760
■ Configuring a Distributed Cache	763
■ Making a Cache Searchable	772
■ Working with Cache Managers	775
■ Working with Caches	783
■ Logging Ehcache Activity	797

What is Ehcache?

Ehcache is a standards-based caching API that is used by Integration Server. Caching enables an application to fetch frequently used data from memory (or other nearby resource) rather than having to retrieve it from a database or other back-end system each time the data is needed.

Ehcache is designed to support multiple deployment configurations, including those that enable you to create very large in-memory caches and share caches across multiple Integration Servers.

Integration Server, and Software AG components that run on Integration Server, use Ehcache to cache data associated with many of their own internal operations. Additionally, Integration Server provides a set of public services (in the `pub.cache` folder in the `WmPublic` package) that developers can use to add caching capabilities to the solutions that they build. For example, a service that frequently reads data from a sales catalog might cache all or part of that catalog to avoid having to query the database each time it needs information about a catalog item. Because accessing data from memory is as much as 1,000 times faster than retrieving it from a database, developers can dramatically improve the performance of a service by caching the data that it uses.

This chapter provides a broad overview of the caching capabilities provided by Ehcache. For a more detailed explanation of these features, see the Ehcache product documentation.

Caching Configurations

A cache on Integration Server occupies a portion of the heap in the Java Virtual Machine (JVM). This portion of a cache is referred to as *on-heap cache*. You can optionally extend a cache beyond the heap to the following locations, commonly referred to as *tiers*:

- Local Disk Storage
- BigMemory (Terracotta license required)
- Terracotta Server Array (Terracotta license required)

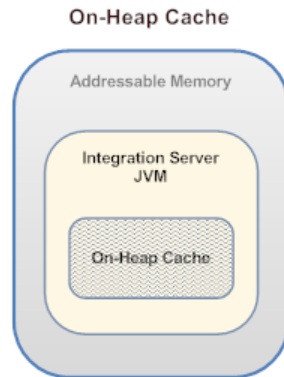
The way in which a cache is divided between on-heap memory and the other tiers is determined by configuration properties that you specify when you create the cache.

The tiered approach enables you to extend a cache beyond the size constraints imposed by the heap. Additionally, when you extend a cache to the Terracotta Server Array, the cache can be simultaneously shared by multiple Integration Servers.

Developers who develop services that use Ehcache do not need to know whether the cache they are using resides only on the heap or whether it extends to one of the other tiers. They simply need to code their service to use a particular cache. Whether that cache extends to disk, BigMemory, or a Terracotta Server Array is determined by configuration properties that are assigned to the cache itself. Consequently, developers do not need to change their code when the caching requirements for their applications change.

On-Heap Cache

On-heap cache is the portion of a cache that resides within the heap of the JVM where the Integration Server is running.



On heap-cache is fast, but because it resides within the heap space, it is subject to the garbage collection process. For large caches, the garbage collection process can cause lengthy interruptions during the execution of a service. If the heap is large, the interruption might be multiple seconds.

Note:

The amount of memory that you can use for on-heap caching depends on the amount of memory you can allocate to the JVM heap. This limit varies according to the JVM, the machine, and the operating system that you are using.

Out of the box, Integration Server supports the use of on-heap caches. You do not need an additional license to create and use an on-heap cache.

For more information about using an on-heap cache with Integration Server, see [“Configuring an On-Heap Cache” on page 758](#).

Local Disk Store

You can optionally associate a local disk store with a cache. A local disk store is a designated directory to which Integration Server writes objects that are placed in cache. You can configure a cache to use a local disk store in one of two ways:

- You can use it to extend the on-heap cache so that cached objects overflow to the local disk store when the on-heap cache becomes full.
- OR --
- You can use it to persist all cached objects to disk so that cached objects are retained when the Integration Server is restarted or the cache manager is reinitialized.

Note:

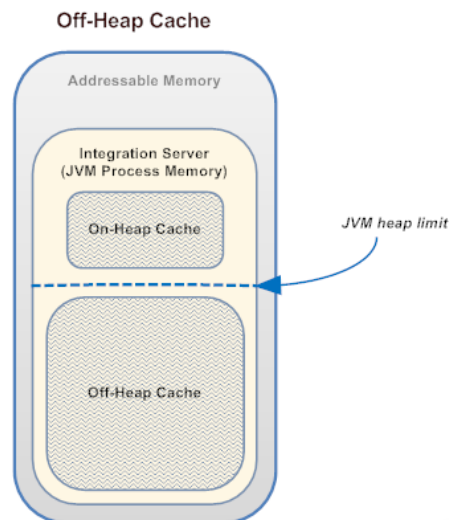
Persisting a cache to the local disk store does not guarantee that elements in cache will be successfully recovered when an Integration Server or a cache manager restarts. If Integration Server or a cache manager is not shut down gracefully, the disk store might not be in a state from which the elements can be recovered. If you require guaranteed persistence of a cache, you must use a Terracotta Server Array.

Note:

You can only use a local disk store with a *local cache*. A local cache is any Ehcache-based cache that resides wholly on the machine where Integration Server is running and does not extend to the Terracotta Server Array.

BigMemory

BigMemory enables you to extend a cache beyond the local JVM heap. When you configure a cache to use BigMemory, a portion of the cache resides within the heap and a portion resides “off heap.” The off-heap portion of the cache resides within the JVM process memory, but outside of the JVM heap.



Because a BigMemory cache resides off heap, it is not subject to the JVM garbage collection process and, consequently, it performs more predictably and consistently. For large caches, BigMemory generally delivers better performance than a heap-based cache. Moreover, from an administrative point of view, caches that use BigMemory are easier to manage, because they do not need to be tuned for garbage collection.

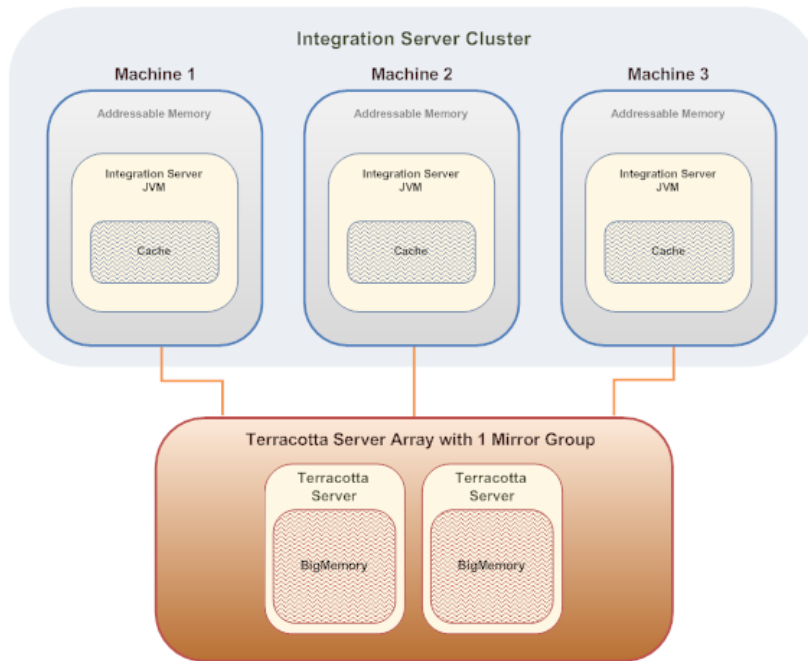
Using BigMemory, you can create much larger caches than with on-heap memory alone. Unlike on-heap caches, which are constrained by the limitations on heap size, BigMemory enables you to use up to a terabyte of memory for caching.

To use BigMemory, your Integration Server must be equipped with a Terracotta license from Software AG.

For more information about configuring a cache to use BigMemory, see [“Configuring a BigMemory Cache” on page 760](#).

Terracotta Server Array

A Terracotta Server Array enables you to create *distributed caches*. A distributed cache can be shared by multiple Integration Servers. Clustered Integration Servers, for example, use a distributed cache to hold data that all the Integration Servers in the cluster need to share.



When you create a distributed cache, a complete copy of the cache resides on a Terracotta Server Array. A Terracotta Server Array consists of one or more Terracotta Servers. It can cache vast amounts of data (e.g., multiple terabytes). The data in the cache is spread across the Terracotta Servers using a technique called “striping.” A Terracotta client on the Integration Server manages the interaction between the Integration Server and the cache on the Terracotta Server Array.

The Integration Servers that share a cache connect to the Terracotta Server Array to put data into the cache and retrieve data from it. Each Integration Server keeps a small portion of the cache locally, which serves as a “hot set” of recently used data and reduces trips to the Terracotta Server Array.

To use a distributed cache, your Integration Server must be equipped with a Terracotta license from Software AG.

For more information about using a distributed cache with Integration Server, see [“Configuring a Distributed Cache”](#) on page 763.

Understanding Caches and Cache Managers

With Ehcache, a cache holds elements that are represented as key-value pairs. The key and its value are both Java objects. Keep the following points in mind about key-value pairs in a cache:

- A key can be of any type of Java object *except* an IS document.

- A value can be of any type of Java object.
- If you are caching data to BigMemory or a Terracotta Server Array, or you write the cache to the local disk store, the objects must be serializable.

Note:

IData implementors are serializable and therefore can be included in a cache as a value. However, the IData itself must contain serializable objects. If the IData to be cached includes non-serializable objects such as streams, XML nodes, or custom objects that are not serializable, Terracotta throws an exception when attempting to write to the cache.

An element that is placed in cache is identified by its key. For example, when Integration Server caches a session object, it uses the session ID as the key. Similarly, when developers create services that put objects into a cache, they select an appropriate identifier to use as the key to retrieve the objects from cache. For example, an application that caches customer objects might derive its key from the customer account number.

A cache has an associated *cache manager*. A cache manager serves as a container for grouping a set of caches into one “management unit” that you can start and shut down together.

The caches that you associate with a cache manager can have different characteristics and configurations. For example, cache manager XYZ might include one cache that resides entirely on the heap and another that extends to BigMemory.

Although there are many properties that you can configure on a cache-by-cache basis, there are certain properties that you assign at the cache manager level. These include:

- **The location of the local disk store.** All of the caches associated with a cache manager use the same disk store location. If you want two caches to use different local disk stores, then you must create those caches in different cache managers.
- **The location of the Terracotta Server Array .** All of the distributed caches associated with a cache manager use the same Terracotta Server Array. If you want two caches to use two different Terracotta Server Arrays, then you must create those caches in different cache managers.

The cache managers that you create on an Integration Server are referred to as *public cache managers*. Public cache managers (and the caches that they contain) can be used by flow services that your developers build.

For more information about creating and configuring public cache managers and caches, see [“Working with Cache Managers” on page 775](#) and [“Working with Caches” on page 783](#).

System Caches

Integration Server and other Software AG products use Ehcache in many of their own internal processes. The caches that they use are called *system caches* and they belong to *system cache managers*.

System cache managers are identified by names that begin with the following characters:

SoftwareAG.

For example,

```
SoftwareAG.IS.Core
```

When you use the Integration Server Administrator to display the list of cache managers that exist on Integration Server, it displays system cache managers and public (user-defined) cache managers in separate lists.

For a list of the system caches used by Integration Server, see *Using Terracotta with webMethods Products*. This document includes a description of the system caches that Integration Server and other Software AG components use and provides sizing information for them.

Cache Manager Configuration Files

Every cache manager has an associated XML file that specifies the configuration parameters for the cache manager and its caches. (In Ehcache documentation, this file is often referred to as the *ehcache configuration file* or the *ehcache.xml* file.)

Integration Server maintains the configuration files for all cache managers (both system cache managers and public cache managers) in the following directory:

```
Integration Server_directory \instances\instance_name\config\Caching
```

When you create a new cache manager using the Integration Server Administrator, Integration Server creates a configuration file for the cache manager and saves the file in *Integration Server_directory \instances\instance_name\config\Caching* directory. Integration Server gives the configuration file the same name as the cache manager, except that it uses dashes (-) instead of any periods that appear in the name. For example the configuration file for a cache manager named `my.cache.manager` would be given the name `my-cache-manager.xml`.

When you start Integration Server, it looks for configuration files in the *Integration Server_directory \instances\instance_name\config\Caching* directory. When it finds a valid configuration file, it initializes the cache manager that is defined by the file and starts the caches that belong to the cache manager.

If Integration Server is not able to initialize a cache manager during its startup process, (if, for example, the cache manager configuration file is invalid or the file defines a type of cache that you are not licensed to use), it reports the error in the server log.

Note:

It is possible for developers to use the Ehcache API to create cache managers and caches without adding a corresponding configuration file to the *Integration Server_directory \instances\instance_name\config\Caching* directory. However, this practice is not recommended on Integration Server. A cache manager whose configuration file is not present in the *Integration Server_directory \instances\instance_name\config\Caching* directory will not be registered properly on Integration Server and cannot be administered using the Integration Server Administrator.

Specifying the Parameters for a Cache

You create caches and edit their parameters using the Integration Server Administrator. When you create a cache, Integration Server Administrator adds the cache to the appropriate cache manager configuration file. When you edit the parameters of a cache, Integration Server Administrator updates the parameters in the cache manager configuration file accordingly.

Important:

While it is possible to modify the parameters for a cache by manually editing a cache manager configuration file, doing this is not recommended. You should always use Integration Server Administrator to edit the parameters of a cache manager or a cache. The only time you should manually edit the configuration file is to set parameters that are not exposed in the Integration Server Administrator user interface (which is rarely necessary). For additional precautions about editing a configuration file, see [“Manually Editing a Cache Manager Configuration File” on page 781](#).

Dynamic vs. Non-Dynamic Cache Parameters

A cache has both *dynamic* and *non-dynamic* parameters. When you change a dynamic parameter using Integration Server Administrator, the change is applied to the cache immediately. You do not need to restart the Integration Server or reinitialize the cache manager to put the change into effect. When you change a non-dynamic parameter, the change does not take effect until you reinitialize the cache manager as described in [“Reinitializing a Cache Manager” on page 779](#).

The parameter descriptions in [“Creating a Cache” on page 783](#) indicate which parameters are dynamic and which are not.

Note:

Apart from being dynamic or non-dynamic, the parameters for a distributed cache are also *cache-wide* or *client-specific*. For details about these qualities, see [“Cache-Wide and Client-Specific Parameters for a Distributed Cache” on page 767](#).

Installing, Viewing, and Changing the Terracotta License

The licensing requirements for Terracotta components depend on the type of component you want to use. To use local, on-heap cache, you do not need an additional Terracotta license. The basic license installed with your Integration Server provides this capability. However, to create a cache that uses BigMemory or the Terracotta Server Array, you must have the appropriate Terracotta and Integration Server licenses.

The following table describes the licensing requirements for each Terracotta component.

If you want to use...

Local cache

You must have...

- An Integration Server license. This license is installed with your Integration Server installation.

Note:

If you want to use...	You must have...
BigMemory cache	<p data-bbox="667 260 1385 348">An additional Terracotta license is not required for this component.</p> <ul style="list-style-type: none"> <li data-bbox="667 359 1385 432">■ An Integration Server license. This license file is installed with your Integration Server installation. <li data-bbox="667 453 1385 558">■ A Terracotta license for BigMemory. You add this license to the host machine where Integration Server is running.
Distributed cache	<ul style="list-style-type: none"> <li data-bbox="667 579 1385 684">■ An Integration Server license for distributed cache. This license is installed with your Integration Server installation. <li data-bbox="667 705 1385 816">■ An upgraded Terracotta BigMemory license for the Terracotta Server Array. You add this license to each Integration Server that shares the distributed cache.
Clustering	<ul style="list-style-type: none"> <li data-bbox="667 837 1385 942">■ An Integration Server license for clustering. This license is installed with your Integration Server installation. <li data-bbox="667 963 1385 1075">■ A Terracotta BigMemory license for the Terracotta Server Array. You add this license to each Integration Server in the cluster.
Clustering with Terracotta Server Array and off-heap storage	<ul style="list-style-type: none"> <li data-bbox="667 1096 1385 1201">■ An Integration Server license for clustering. This license is installed with your Integration Server installation. <li data-bbox="667 1222 1385 1331">■ An upgraded Terracotta BigMemory license for the Terracotta Server Array. You add this license to each Integration Server in the cluster.

Determining if You Have a Terracotta License

If you are not sure whether you have a Terracotta license, you can view licensed components for your installation from the **Settings > Licensing** screen in the Integration Server Administrator. If you have a Terracotta license installed, it is listed here. You can click **Licensing Details** to view the Terracotta components for which you are licensed. If you do not have a Terracotta license, you can obtain one from Software AG.

Adding a Terracotta License

The Terracotta license is a file named `terracotta-license.key`. It contains the license information for all of your Terracotta components. You add this file to Integration Server by placing it into the `Software AG_directory \common\conf` directory of the machine on which Integration Server runs.

The Integration Server Administrator checks this directory for licenses and displays the information on the Licensing screen.

If the license file is located in a different directory, you can add it to Integration Server using the Integration Server Administrator.

➤ **To add a Terracotta license key using Integration Server Administrator**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Licensing**.
3. Click **Licensing Details**.
4. Click **Edit Licensing Detail**.
5. In the **Terracotta License File** field, enter the fully qualified file name for the license key file you obtained from Software AG.
6. Click **Save Changes**. Integration Server Administrator updates the **Terracotta License File** field to reflect the file name that you entered.
7. Restart Integration Server.

Note:

You must restart the Integration Server after adding, removing, or changing a Terracotta license.

Configuring an On-Heap Cache

Perform the following general steps to create an on-heap cache. An on-heap cache is maintained entirely within the JVM heap. On-heap caches are fast, but they are subject to the JVM garbage collection process. For more information about on-heap caches, see [“On-Heap Cache” on page 751](#).

Step	Description
1	Review the points listed in “Considerations for Configuring On-Heap Cache” on page 759 .
2	Decide how much memory you need for the cache and adjust the heap size of your JVM if necessary. For information about changing the heap size used by Integration Server, see “Changing the JVM Heap Size Used by Integration Server ” on page 130 .
3	If the cache manager to which you want to add the cache does not already exist, create it. For information about creating a cache manager, see “Creating a Cache Manager” on page 775 .

Step	Description
4	<p>Create the cache using the procedure in “Creating a Cache” on page 783. When you create the cache, be sure to do the following:</p> <ul style="list-style-type: none"> ■ Complete the parameters in the Cache Configuration and Cache Configuration Advanced Settings sections. ■ <i>Do not</i> enable the Distributed option in Distributed Cache Configuration section. ■ <i>Do not</i> enable the Overflow to Off-heap option in BigMemory section.
5	To make the cache available for use, enable the cache as described in “Enabling a Cache” on page 795 .

Considerations for Configuring On-Heap Cache

When you create an on-heap cache, keep the following points in mind:

- The heap space is shared by all processes running in the JVM. When you add an on-heap cache to Integration Server, you may need to increase the size of the heap to ensure that it is large enough to satisfy the heap requirements of the new cache and other consumers of the heap. These include:
 - Other on-heap caches that exist on the Integration Server (including system caches).
 - All classes, class variables and instance variables used by Integration Server and the services and triggers that are deployed on it.
 - OSGI processes that are running in the same JVM as Integration Server (such as the webMethods Event Server).

For sizing information about the system caches that Integration Server and other Software AG products use, see *Using Terracotta with webMethods Products* on the Software AG documentation website.

- To estimate the amount of the heap that a cache will consume, multiply the size of objects that you want to cache by the number of those objects that you want to keep in cache.

$$\text{objectSizeInBytes} * \text{numObjectsToCache} = \text{heapConsumedByCache}$$

- Very large on-heap caches (which require a large heap size) increase the time it takes for the JVM to perform a full garbage collection. When a full garbage collection occurs, all threads in the JVM are paused, which can cause noticeable delays in response times.

When sizing an on-heap cache, you must weigh the benefits of keeping a large amount of data in memory against the periodic interruptions that this imposes on the Integration Server. For additional information about the effect of garbage collection on a cache, see Ehcache product documentation for 2.8 at <http://ehcache.org/documentation>.

- If garbage collection becomes a significant problem with the size of your on-heap cache, consider maintaining a smaller on-heap cache and enabling **Overflow to Disk** or **Overflow to Off-Heap** (BigMemory license required) to extend the cache without increasing its on-heap requirements.
- Make sure your machine is equipped with sufficient memory to support the heap requirements of your JVM and the memory requirements of other applications that run on the machine. Insufficient memory will cause paging.
- If you enable the **Overflow to Disk** or **Persist to Disk** option, your cache must contain only serializable keys and values. If a service attempts to put a non-serializable object into a cache that overflows or is persisted to disk, the service will receive a runtime exception.

Configuring a BigMemory Cache

Perform the following general steps to create a BigMemory cache. A BigMemory cache occupies both on-heap and off-heap memory. (For this reason, it is sometimes referred to as an *off-heap cache*.) When you configure a BigMemory cache, you specify the number of elements that you want the Integration Server to maintain on-heap. You also specify how much off-heap memory the cache can use. A BigMemory cache is fast and can be very large. The off-heap portion of the cache is not subject to JVM garbage collection. For more information about BigMemory caches, see “[BigMemory](#)” on page 752.

To create a cache that uses BigMemory, perform the following general steps:

Step	Description
1	Verify that the Integration Server is equipped with a license to use BigMemory and check how much BigMemory the license authorizes. For procedures, see “ Installing, Viewing, and Changing the Terracotta License ” on page 756.
2	If you have not done so already, allocate direct memory to Integration Server using the procedure in “ Allocating Direct Memory Space to Integration Server ” on page 761.
3	Review the points listed in “ Considerations for Configuring BigMemoryCache ” on page 762.
4	Decide how much memory you need for the on-heap portion of the cache and adjust the Java heap size if necessary. For information about changing the heap size used by Integration Server, see “ Changing the JVM Heap Size Used by Integration Server ” on page 130.
5	If the cache manager to which you want to add the cache does not already exist, create it. For information about creating a cache manager, see “ Creating a Cache Manager ” on page 775.
6	Create the cache using the procedure in “ Creating a Cache ” on page 783. When you create the cache, be sure to do the following:

Step	Description
	<ul style="list-style-type: none"> <li data-bbox="479 262 1373 399">■ In the Maximum Elements in Memory field, specify the number of elements that you want to maintain in on-heap memory. (To avoid performance issues, always set Maximum Elements in Memory to a value greater than 100 elements.) <li data-bbox="479 430 1373 525">■ Enable Overflow to Off-Heap and, in the Max Off-Heap field, specify the amount of off-heap memory that you want to allocate to this cache.
7	To make the cache available for use, enable the cache as described in “Enabling a Cache” on page 795 .

Allocating Direct Memory Space to Integration Server

To use BigMemory with Integration Server, you must allocate *direct memory* space to the JVM in which Integration Server is running.

The amount of direct memory space that is given to Integration Server is determined by the `wrapper.java.additional.n=-XX:MaxDirectMemorySize` property in the `custom_wrapper.conf` file. By default, the `wrapper.java.additional.n=-XX:MaxDirectMemorySize` property is not set when Integration Server is installed.

When you use BigMemory, you must set the `wrapper.java.additional.n=-XX:MaxDirectMemorySize` property to the amount of memory that you need to hold your off-heap caches.

Note:

To allocate direct memory space to Microservices Runtime, add `-XX:MaxDirectMemorySize` to `JAVA_CUSTOM_OPTS` in `Integration Server_directory/bin/setenv.bat(sh)`. For information about passing Java system properties to Microservices Runtime, see *Developing Microservices with webMethods Microservices Runtime*.

➤ To allocate direct memory space to Integration Server

1. On the machine where Integration Server is installed, navigate to the following folder:

```
Software AG_directory \profiles\IS_instance_name\configuration
```

2. In a text editor, open `custom_wrapper.conf`.
3. Add a the `wrapper.java.additional.n=-XX:MaxDirectMemorySize` property to specify the amount of BigMemory required to accommodate all of your caches that use BigMemory. (This amount is limited by your Terracotta license, the amount of physical memory on your system, and how much of that memory is needed by the operating system and other applications running on the machine.)

For example, if you determine that you need 500 megabytes of BigMemory and you are running Integration Server under Windows, you would add the `wrapper.java.additional` property in the `custom_wrapper.conf` file to look like this:

```
wrapper.java.additional.n--XX:MaxDirectMemorySize=500M
```

Where *n* is the next unused sequential number for the `wrapper.java.additional` properties in the file.

Note:

You can express the value of this parameter using 'm' or 'M' for megabytes or using 'g' or 'G' for gigabytes.

4. Save and close `custom_wrapper.conf`.
5. Restart Integration Server.

Considerations for Configuring BigMemoryCache

When you create a cache that uses BigMemory, keep the following points in mind:

- A BigMemory cache accepts only serializable keys and values. If a service attempts to put a non-serializable object into the cache at run time, the service will receive an exception.
- Allocate at least 100 elements to the on-heap portion of cache. Smaller on-heap cache sizes do not perform efficiently. Integration Server will log a warning message if a BigMemory cache is initialized with fewer than 100 elements allocated to the on-heap portion of the cache.
- Allocate at least 128 megabytes to the off-heap portion of the cache.
- Be aware that not all of the direct memory that you allocate to the JVM is available for off-heap caches. You must reserve 32 megabytes for non-cache use by BigMemory. (In other words, the amount of memory available for off-heap caching is *MaxDirectMemorySize* minus 32M.)
- Before you specify the off-heap memory for the new cache, check how much BigMemory is already allocated by off-heap caches that exist on the Integration Server. The off-heap memory that you specify for this cache must fit within whatever direct memory remains (minus the 32 megabytes of that memory is reserved for non-cache use, as mentioned above).

To determine how much BigMemory is already used by existing caches, examine the **BigMemory** column on the **Settings > Caching > CacheManagerName** page for each cache manager.

- Integration Server allocates the off-heap portion of a cache (the amount of memory specified in the **Maximum Off-Heap**) when it instantiates the cache. If it cannot obtain the amount of memory specified in **Maximum Off-Heap**, the Integration Server writes an error message to the server log, places a stack trace in the error log, and does not enable the cache.
- The off-heap memory that you allocate to a cache actually includes a copy of the elements that reside in on-heap cache. Keep this point in mind when you specify the **Maximum Off-Heap** parameter for the cache. For example, if you set **Maximum Off-Heap** to 128 MB and your

elements are approximately 1 KB each, your cache will be able to hold about 128,000 elements in off-heap memory. However, this number includes copies of the on-heap elements. If the size of your on-heap cache is 2,000 elements, then the off-heap cache will hold copies of those 2,000 elements, plus 126,000 additional elements.

- If you enable the **Overflow to Disk** option and you want to limit the number of elements that can be written to disk using the **Maximum Entries Local Disk** parameter, set **Maximum Entries Local Disk** to a value that is greater than the number of elements that the amount of memory in **Maximum Off-Heap** can hold. For example, if **Maximum Off-Heap** will hold about 128,000 elements and you want to allow an additional 50,000 elements to overflow to disk, set **Maximum Entries Local Disk** to 178,000 elements.
- For other considerations relating to BigMemory caches, see the BigMemory Go product documentation for 4.1 at www.terracotta.org/documentation.

Configuring a Distributed Cache

Perform the following general steps to create a distributed cache. A distributed cache resides on a Terracotta Server Array and can be shared with other Integration Servers. Integration Server keeps a portion of the distributed cache in its on-heap memory so that it does not have to access the Terracotta Server Array every time it needs to fetch data from the cache. When you configure a distributed cache, you specify the number of elements that you want the Integration Server to maintain locally in the on-heap portion of the cache. For more information about distributed caches, see “[Terracotta Server Array](#)” on page 753.

Step	Description
1	<p>If you have not already installed and configured your Terracotta Server Array, do this first. For procedures, see the product documentation for the Terracotta Server Array.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: The Terracotta Server Array must be running a version of Terracotta that is compatible with the Terracotta client libraries installed on your Integration Server. To check which versions of the Terracotta Server Array are compatible with your Integration Server, see the <i>Terracotta Compatibility with other Software AG Products</i> on the Software AG documentation website at http://documentation.softwareag.com.</p> </div>
2	<p>If you have not done so already, edit the tc-config file on the Terracotta Server Array and add the parameters that Integration Server requires. For procedures, see “Configuring tc-config.xml on the Terracotta Server Array” on page 765.</p>
3	<p>Review the points listed in “Considerations for Configuring a Distributed Cache” on page 766.</p>

Step	Description
4	Verify that Integration Server is equipped with a license to use a Terracotta Server Array. For procedures, see “Installing, Viewing, and Changing the Terracotta License” on page 756.
5	Decide how much memory you will use for the on-heap portion of the cache on Integration Server and adjust the Java heap size if necessary. For information about changing the heap size used by Integration Server, see “Changing the JVM Heap Size Used by Integration Server” on page 130. Note: The size of the on-heap portion of a distributed cache must be the same on all Integration Servers that use the cache.
6	If the cache manager to which you want to add the cache does not already exist, create it. For information about creating a cache manager, see “Creating a Cache Manager” on page 775. Note: Be sure that the cache manager that you are using is configured to use the Terracotta Server Array.
7	Create the cache using the procedure in “Creating a Cache” on page 783. When you create the cache, be sure to do the following: <ul style="list-style-type: none">■ Enable the Distributed setting in the Distributed Cache Configuration section of the page.■ In the Maximum Elements in Memory field, specify the number of elements that you want to maintain in the on-heap memory at the Integration Server. Note: Although Integration Server permits you to set Maximum Elements in Memory 0, which indicates that the number of elements is restricted only by the availability of resources, Software AG strongly recommends that you always specify an upper limit. Unlimited caches grow indefinitely and can cause performance problems or system failures if they exhaust the available resources on the machine.■ In the Maximum Entries in Cache field, specify the maximum size (in number of elements) to which this cache can grow on the Terracotta Server Array.
8	Verify that the parameters for the cache are set as you want them and enable the cache as described in “Enabling a Cache” on page 795.

Step	Description
	If this is the first Integration Server on which you have created the distributed cache, the Integration Server creates the distributed cache on the Terracotta Server Array during this step.
9	If other Integration Servers will use this distributed cache, repeat steps 4 through 8 on each of them.

Note:

Instead of manually specifying the new cache on each of the other Integration Servers in step 7, you can optionally create the cache by copying the cache manager configuration file from this Integration Server to the others. You can do this by 1) using webMethods Deployer to deploy the cache manager from this Integration Server to the others or 2) manually copying the cache manager configuration file from this Integration Server to the others. For information about using webMethods Deployer, see the *webMethods Deployer User's Guide*. For information about copying a cache manager configuration file to another Integration Server, see ["Adding a Cache Manager Configuration File to Integration Server"](#) on page 780.

Configuring tc-config.xml on the Terracotta Server Array

Integration Server requires that you set properties in the tc-config.xml file to configure the Terracotta Server Array. After you set the required properties and start Integration Server, Integration Server downloads tc-config.xml to obtain the information necessary to connect to the Terracotta Server Array.

You specify the host servers for clustering or caching configuration with the **Terracotta Server Array URLs** parameter in Integration Server Administrator. When you start public cache managers that utilize distributed caches or a clustered Integration Server using distributed system caches, Integration Server uses the **Terracotta Server Array URLs** parameter to download tc-config.xml from the first specified Terracotta Server Array server to which it is able to connect. Using the settings in tc-config.xml, Integration Server connects to the Terracotta Server Array to begin using distributed caches. For more information about setting the **Terracotta Server Array URLs** parameter, see ["Working with Cache Managers"](#) on page 775 (for distributed caches) or the *webMethods Integration Server Clustering Guide* (for Integration Server using Terracotta for clustering).

Configure the settings in tc-config.xml after you install the Terracotta Server Array, but *before* you start either the distributed cache on Integration Server or a clustered Integration Server that uses distributed system caches.

➤ To configure the parameters in tc-config.xml

1. On the Terracotta Server Array host server, open tc-config.xml with a text editor.

Note:

By default, a Terracotta server expects to find the tc-config file in the *TerracottaHome/server/bin* folder. If tc-config.xml does not already exist on the Terracotta Server Array host server you must create it.

2. Specify the following element contents:

Note:

The following parameters are *required* for Integration Server clusters using distributed system caches. You must specify the values for the element contents exactly as shown.

- a. In the tc-properties element, specify the property attributes as follows:

```
<tc-properties>
<property name="ehcache.storageStrategy.dcv2.perElementTTITTL.enabled"
value="true"/>
</tc-properties>
```

The ehcache.storageStrategy.dcv2.perElementTTITTL.enabled property determines whether the Terracotta server supports per element time-to-live and time-to-idle expiration. A true value, which is required for Integration Servers in a cluster, indicates that expiration is per element.

- b. In the clients element, specify the content of the logs child element as follows:

```
<clients>
<logs>%(com.softwareag.tc.client.logs.directory)</logs>
</clients>
```

This element defines the directory to which Integration Server writes log entries. For more information about logging, see [“Logging Ehcache Activity” on page 797](#).

3. When you are finished making your changes, save and close the file.

Note:

For more information about configuring Terracotta Server Array see the Terracotta product documentation. The *Using Terracotta with webMethods Products* contains a sample tc-config.xml file.

Considerations for Configuring a Distributed Cache

Keep the following points in mind when configuring a distributed cache.

- A distributed cache accepts only serializable keys and values. If a service attempts to put a non-serializable object into the cache at run time, the service will receive an exception.
- A distributed cache uses the disk store on the Terracotta Server Array, not on the Integration Server. You cannot configure the portion of distributed cache that resides on the Integration Server to overflow or persist to disk. When you create a distributed cache using Integration Server Administrator, the **Overflow to Disk** or **Disk Persistent** settings are disabled.
- An Integration Server references a cache on the Terracotta Server Array using a fully qualified cache name. The fully qualified name of a cache consists of the name of the cache manager

and the name of the cache. For example, the fully qualified name of a cache called “OrderDetails” in a cache manager called “Orders” is “Orders.OrderDetails.” If you have multiple Integration Servers that share a distributed cache, be sure that they all use the same fully qualified name for the cache.

- To create a distributed cache that will be shared by multiple Integration Servers, you first create and enable the cache on one of the Integration Servers. This step registers the distributed cache on the Integration Server and also creates the cache on the Terracotta Server Array. Then you add the distributed cache to each additional Integration Server that will use the cache. When you enable the distributed cache on these Integration Servers, they will see that the cache already exists on the Terracotta Server Array and will begin using it.
- Certain parameters for a distributed cache are *cache-wide*, meaning that when you change the parameter on one Integration Server, it affects all Integration Servers that use the cache. For information about which parameters are cache-wide, see [“Cache-Wide and Client-Specific Parameters for a Distributed Cache” on page 767](#).
- For a Terracotta Server Array you can set failover behavior to consistency instead of availability (the default). For information, see the section on failover tuning for guaranteed consistency in the *BigMemory Max Administrator Guide*.

Cache-Wide and Client-Specific Parameters for a Distributed Cache

A distributed cache has both *cache-wide* parameters and *client-specific* parameters.

Cache-Wide Parameters

The cache-wide parameter settings for a distributed cache are maintained on the Terracotta Server Array and affect all of the Integration Servers that use the cache. When you change a cache-wide parameter setting on one Integration Server, the change is applied to all Integration Servers that use the cache.

The following table identifies the cache-wide parameters and indicates whether they affect the cache on the Terracotta Server Array, the local portion of the cache on the Integration Servers, or both. As noted in the table, all cache-wide parameter settings are dynamic, meaning that changes you make to them take effect immediately. You do not need to reinitialize the cache manager or restart the Terracotta Server Array to put the changes into effect.

Cache-Wide Parameter	Applies to...	Dynamic?
Max Elements in Memory	All Integration Servers that use the cache. Does not affect the size of the cache on the Terracotta Server Array itself.	Yes
Eternal	The cache on the TSA and all Integration Servers that use the cache.	Yes

Cache-Wide Parameter	Applies to...	Dynamic?
Time to Live	The cache on the TSA and all Integration Servers that use the cache.	Yes
	<p>Important: <i>Do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, first start the cache manager by following the procedure in “Starting a Cache Manager” on page 778.</p>	
Time to Idle	The cache on the TSA and all Integration Servers that use the cache.	Yes
	<p>Important: <i>Do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, first start the cache manager by following the procedure in “Starting a Cache Manager” on page 778.</p>	
Maximum Entries in Cache	The cache on the TSA. Does not affect the cache on the Integration Servers.	Yes
	<p>Important: <i>Do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, first start the cache manager by following the procedure in “Starting a Cache Manager” on page 778.</p>	

Client-Specific Parameters

Client-specific parameters affect the way in which a given Integration Server interacts with a distributed cache. When you change a client-specific parameter, the new setting affects only the Integration Server on which you make the change.

As noted in the following table, some client-specific parameters are dynamic and others are non-dynamic. Except where noted, non-dynamic parameters require that you shut down the cache manager (as described in [“Shutting Down a Cache Manager” on page 777](#)). After you make your changes, reinitialize the cache manager (as described in [“Reinitializing a Cache Manager” on page 779](#)) to put the change into effect. For additional information about the following parameters, see the parameter descriptions in [“Creating a Cache” on page 783](#).

Client-Specific Parameter	Dynamic?
Clear on Flush	No
Copy on Read	No
<p>Important: <i>Do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, follow the procedure in “Modifying Settings for a Distributed Cache” on page 794.</p>	
Copy on Write	No
Synchronous Writes	No
Consistency	No
<p>Important: <i>Do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, follow the procedure in “Modifying Settings for a Distributed Cache” on page 794.</p>	
Memory Store Eviction Policy	Yes
Enable High Availability	No
Timeout	Yes
Timeout Behavior	Yes
Immediate Timeout When Disconnected	Yes
Maximum Entries Local Disk	Yes

Note:

Technically speaking, the three parameters related to high availability (**Timeout**, **Timeout Behavior**, and **Immediate Timeout When Disconnected**) behave dynamically only when the **Enable High Availability** option is enabled. However, on Integration Server, **Enable High Availability** is always enabled for a distributed cache and thus, these three parameters always behave dynamically.

The Rejoin Behavior of a Distributed Cache

The **Rejoin** parameter enables a cache manager to automatically reconnect to a Terracotta Server Array from which it has become disconnected. This option eliminates the need to manually reinitialize a cache manager to re-establish a broken connection to the Terracotta Server Array.

Integration Server automatically enables the **Rejoin** parameter for all cache managers whose **Terracotta Server Array URLs** parameter is set. You cannot disable the **Rejoin** setting for such cache managers.

Note:

When a cache manager rejoins a Terracotta Server Array, it joins the array as though it were a new client. This means that Integration Server clears any data that it currently holds locally for the cache and resynchronizes its on-heap store with data from the distributed cache on the Terracotta Server Array.

Note:

If your Integration Server becomes disconnected from the Terracotta Server Array frequently, you might encounter OutOfMemory errors. These errors can occur because certain class definitions are reloaded each time the Integration Server rejoins the Terracotta Server Array. If many rejoins occur in a short period of time, disused class definitions can fill the permanent generation space on the heap before they can be garbage collected. For information about this error condition and how you can adjust the size of the permanent generation area to avoid it, see “Avoiding OOME from Multiple Rejoins” in the Terracotta product documentation.

For more information about the rejoin feature of a distributed cache, see “Rejoin” in the BigMemory Max product documentation for 4.1 at www.terracotta.org/documentation.

For more information about the **Rejoin** parameter on Integration Server, see “[Creating a Cache Manager](#)” on page 775.

The Nonstop Behavior of a Distributed Cache

The **Enable High Availability** parameter places a distributed cache in *nonstop mode*. Nonstop mode enables the Integration Server to take a prescribed action if an operation on the distributed cache does not complete within a specified timeout period (for example, if the Terracotta Server Array is unresponsive because it is busy or if the network connection is down).

When you create a distributed cache with the Integration Server Administrator, it automatically enables the **Enable High Availability** parameter. You cannot disable this option.

Because distributed caches on Integration Server operate in nonstop mode, you must configure the following parameters when you create a distributed cache. These parameters specify how long you want Integration Server to wait for a response from the Terracotta Server Array, and which steps you want it to take when a cache operation takes longer than the specified time.

Parameter	Description
Timeout	Specifies the amount of time (in milliseconds) that must elapse before Integration Server determines that it cannot reach the Terracotta Server Array and performs the time-out measure specified in Timeout Behavior .

Parameter	Description
Timeout Behavior	<p>Specifies which of the following steps you want Integration Server to take in order to complete a cache operation when the Terracotta Server Array does not respond within the amount of time specified in Timeout.</p> <ul style="list-style-type: none"> ■ exception - Return a NonStopCacheException. ■ noop - Return a null for “get” operations. Ignore operations that modify elements in the cache (for example, “put,” “replace,” and “remove” operations). ■ localReads - Return elements that reside in the local cache in response to “get” operations. Ignore operations that modify elements in the cache (for example, “put,” “replace,” and “remove” operations).
Immediate Timeout When Disconnected	<p>Specifies whether Integration Server should automatically time out all subsequent operations for this cache when Integration Server recognizes that it has become disconnected from the Terracotta Server Array.</p> <p>If the Immediate Timeout When Disconnected option is enabled when Integration Server becomes disconnected, Integration Server immediately times out all subsequent operations on the cache and performs the time-out measure specified in Timeout Behavior. This option stops Integration Server from waiting for every operation to timeout when it is in the disconnected state. The Integration Server will begin processing cache operations normally again after it successfully rejoins the Terracotta Server Array.</p>

For more information about the nonstop features of a distributed cache, see the BigMemory Max product documentation for 4.1 at www.terracotta.org/documentation.

For more information about setting these parameters on Integration Server, see [“Creating a Cache” on page 783](#).

About the Maximum Entries in Cache Value

Beginning with Integration Server version 10.0 and in fixes to earlier versions of Integration Server, the **Maximum Entries in Cache** value must be set and must be set to a value greater than 0. Previously, Integration Server allowed a value of 0 which indicated that the number of entries was restricted only by the availability of resources. However, unlimited caches grow indefinitely and can cause performance problems or system failures if they exhaust the available resources on the machine.

To prevent performance problems and an OutOfMemoryException, Integration Server requires that system and public distributed caches have **Maximum Entries in Cache** set to a value greater than 0:

- When creating or editing a public distributed cache created in a version of Integration Server prior to 10.0, Integration Server Administrator validates the **Maximum Entries in Cache** value and prompts for a correction if the **Maximum Entries in Cache** value is left blank or is set to 0 (zero). For an existing public distributed cache created prior to version 10.0, Integration Server allows **Maximum Entries in Cache** to be 0 until the cache is edited in some way.
- When editing a system distributed cache, Integration Server Administrator validates the **Maximum Entries in Cache** value and prompts for a correction if the **Maximum Entries in Cache** value is left blank or is set to 0 (zero). For a system distributed cache, Integration Server allows **Maximum Entries in Cache** to be 0 until the cache is edited in some way.
- For an existing system distributed cache used in an Integration Server cluster, Integration Server sets the value of **Maximum Entries in Cache** to the value specified in new server configuration property `watt.server.cache.maxEntriesInCache`. At startup of Integration Server if clustering is enabled, Integration Server will set the value for distributed system caches to the value set by `watt.server.cache.maxEntriesInCache`. If the server configuration property is set to -1 or 0, Integration Server does not set a default value for **Maximum Entries in Cache**.

Making a Cache Searchable

Integration Server uses the Ehcache search API to build indexes that you can use to look up data in a cache. You can search against predefined indexes of keys, values, or attributes extracted from values.

Some keys and values are directly searchable and can simply be added to the search index as attributes. However, you cannot search some keys or values directly. These require that you first extract the searchable attributes.

Once you make a cache searchable, you can use the `pub.cache:search` service to search the cache. The service returns the search result as an object array. The search results can include:

- Element keys
- Element values
- Predefined attribute values extracted from element values
- Aggregator results, which are results of aggregator functions such as Average, Count, Max, Min, and Sum.

Defining Attributes

When making a cache searchable, you can define the attributes to use in searches. Attributes are extracted from keys or values during searches. You can extract attributes from keys or values using the following methods:

- **Extract by Expression.** Specify expressions based on which attributes are extracted from keys or values. For example,

```
key.name, value.age
```


- **Extract by Class.** Provide the extractor class and optionally, the properties for the extractor class.

For general guidelines for creating an extractor class, see [“Extracting Attributes by Class” on page 773](#). For a sample extractor, see [“Example Extractor” on page 773](#).

Keep the following points in mind when making a cache searchable:

- You cannot configure a system cache to be searchable.
- You cannot make a cache that already contains data searchable.

For more information about making a cache searchable, see [“Creating a Cache” on page 783](#).

Extracting Attributes by Class

To extract attributes using an attribute extractor class, follow the general steps below.

Note:

If you want to search a document (IData) in a cache, you must first create an attribute extractor class that can be used by Ehcache to extract the search attributes. Ehcache uses the extractor class and the provided search attribute information to search the cache when the `pub.cahce:search` service executes.

Step	Description
1	Create an attribute extractor class that is an implementation of <code>net.sf.ehcache.search.attribute.AttributeExtractor</code> . For more information about attribute extractors, see the Terracotta Ehcache product documentation.
2	Place the extractor in a jar file and place the jar file in one of the following directories: <ul style="list-style-type: none"> ■ <code>Integration Server_directory \instances\instance_name \lib\jars\custom</code> ■ <code>Integration Server_directory \instances\instance_name \packages\packageName \code\jars\static</code>
3	Configure the search attributes for the cache to identify: <ul style="list-style-type: none"> ■ The name of attributes extracted by the class ■ The extractor class ■ The properties for the extractor class as key/value pairs, separated by commas (,)

Example Extractor

The following sample shows how to extract an attribute that has a document as the element value in the key/value pair:

```

package com.softwareag.cache.sample ;

import java.util.Properties;
import com.wm.data.IData ;
import com.wm.data.IDataCursor ;
import com.wm.data.IDataUtil ;

import net.sf.ehcache.Element;
import net.sf.ehcache.search.attribute.AttributeExtractor;
import net.sf.ehcache.search.attribute.AttributeExtractorException;
/**
 * This attribute extractor is designed for a Doc Type of the following format
 *
 * PO
 * String number
 * Customer
 * String name
 * String address
 *
 * In our cache search settings we defined 2 search attributes:
 * 1) PONumber - Corresponds to the top level number field
 * 2) CustomerName - Corresponds to the customer\name field
 */
public class IDataAttributeExtractor implements AttributeExtractor {
    private Properties _prop = null;

    /**
     * @param prop Contains the values entered in the "Properties" field for the
     search attribute.
     */
    public IDataAttributeExtractor(Properties prop)
    {
        _prop = prop;
        System.out.println(prop);
    }
    public Object attributeFor(final Element element, final String attributeName)
        throws AttributeExtractorException
    {
        IDataCursor poCursor = null;
        IDataCursor custCursor = null;

        String value = null;
        try
        {
            poCursor = ((IData)element.getObjectValue()).getCursor();

            if (attributeName.equals("PONumber"))
            {
                // The search attribute PONumber corresponds to the top
                // level number field in our stored document.
                value = IDataUtil.getString (poCursor, "number");
            }
            else if (attributeName.equals("CustomerName"))
            {
                // The search attribute CustomerName corresponds to the
                // customer\name field in our stored document.
                IData custIData = IDataUtil.getIData(poCursor, "customer");
                if(custIData == null)

```

```

    {
        //We could not find the customer field in our stored document
        throw new AttributeExtractorException("Unable to find the customer" +
            " field in our element for the " + attributeName + " search attribute.");
    }
    else
    {
        // return the customer->name value
        custCursor = custIData.getCursor();
        value = IDataUtil.getString(custCursor, attributeName);
    }
}
else
{
    // We do not recognize the search attribute
    throw new AttributeExtractorException("Unknown search attribute: " +
attributeName);
}
}
finally
{
    if(poCursor != null) poCursor.destroy();
    if(custCursor != null) custCursor.destroy();
}
return value;
}
}

```

Working with Cache Managers

The procedures in this section describe how to create, view, edit, start, shutdown, reload, and delete cache managers.

Creating a Cache Manager

Use the following procedure to create a cache manager.

➤ To create a cache manager

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Click **Add Cache Manager**.
4. In the Add Cache Manager screen, in the **Name** field, enter a name for the cache manager.

Valid cache manager names:

- Must not be null.

- Must not contain these characters: ? [] / \ = + < > : ; " , * | ^ @ &
 - Must not start with a period (.) or end with a space.
 - Must not begin with the characters SoftwareAG.
 - Must not be a device name in any operating system.
5. If this cache manager will include distributed caches, configure the **Terracotta Server Array URLs** field to specify the Terracotta Server Array on which the caches reside. In this field, enter a comma-separated list of addresses (in *host:port* format), one for each server in the Terracotta Server Array. You can add multiple URLs by using this format:
host1:port,host2:port,...

Note:

When you specify the **Terracotta Server Array URLs** field, the **Rejoin** check box is automatically selected. **Rejoin** indicates that the Integration Server will automatically reconnect to the Terracotta Server Array upon getting disconnected. You cannot disable this option. For more information about the **Rejoin** option, see [“The Rejoin Behavior of a Distributed Cache” on page 769](#).

6. Click **Save Changes**.

When adding a cache manager, Integration Server verifies that the supplied Terracotta Server Array URLs are valid. Integration Server verifies that the provided URLs accept a connection. If the URLs are not valid, then Integration Server throws an `IOException` and does not create the cache manager.

Note: Integration Server does not start the cache manager automatically after creating it. You must start the cache manager by clicking the **Start** link. For information about starting a cache manager see [“Starting a Cache Manager” on page 778](#).

Viewing or Modifying a Cache Manager

Use the following procedure to view or modify a cache manager configuration.

Note:

The modification steps in this procedure apply to public cache managers only. You cannot modify the parameters associated with a system cache manager.

➤ To view or modify a cache manager

1. If you intend to update the **Terracotta Server Array URLs** for a public cache manager, either shut down the cache manager before you proceed or place Integration Server in quiesce mode. You cannot modify this parameter setting while the cache manager is running. For procedures, see [“Shutting Down a Cache Manager” on page 777](#) or [“Quiescing the Server for Maintenance” on page 929](#).

2. Open Integration Server Administrator if it is not already open.
3. In the **Settings** menu of the Navigation panel, click **Caching**.
4. In the cache managers list, click the cache manager that you want to view or modify.
5. If you want to modify the cache manager, click **Edit Cache Manager Configuration** (public cache managers only).
6. Modify the cache manager configuration.
7. Click **Save Changes**.
8. Reload the cache manager to put the changes into effect. For procedures, see [“Reloading a Cache Manager” on page 778](#).

Shutting Down a Cache Manager

When a cache manager is no longer required, you can shut it down to free up resources.

If the cache manager is not connected to a Terracotta Server Array, the caches are cleared when the cache manager is shut down. In such instances, the data in the cache is lost if the cache is not persistent.

If the cache manager is connected to a Terracotta Server Array, the distributed caches associated with the cache manager will not be available to services running on the local Integration Server. However, these caches still exist on the Terracotta Server Array and other Integration Servers can continue to use them.

Note:

This procedure applies to public cache managers only. You cannot shut down a system cache manager.

» To shut down a cache manager

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the **Shutdown** link for the cache manager that you want to shut down.
4. When prompted to confirm that you want to shut down the cache manager, click **OK**.

Starting a Cache Manager

Use the following procedure to start a cache manager. All cache managers are automatically started when Integration Server starts. However, Integration Server does not start a cache manager after you create it. You must use this procedure to start a new cache manager. You can also use this procedure to start a cache manager after it has been shut down.

Note:

This procedure applies to public cache managers only. You cannot manually start a system cache manager.

Integration Server uses the configuration from the cache manager configuration file when starting the cache manager for the first time. When you restart a cache manager, Integration Server uses the configuration information that is cached in the memory.

Note:

When you start a cache manager, the caches within it are enabled automatically.

> To start a cache manager

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. In the **Public Cache Managers** list, click the **Start** link for the cache manager that you want to start.
4. When prompted to confirm that you want to start the cache manager, click **OK**.

Reloading a Cache Manager

Use the following procedure to reload a cache manager without restarting Integration Server. When reloading a cache manager, keep the following points in mind:

- To reload a cache manager, the cache manager must be running.
- When you reload a cache manager, its local caches are deleted and recreated. If a cache is not persistent, the data in it is lost.
- If the cache manager is connected to a Terracotta Server Array, the distributed caches associated with the cache manager will not be available to the services running on this Integration Server during the reload process. However, once the cache manager is reloaded, these caches, and their data, will be available again.


Note:

When you reload a cache manager, the caches within it are enabled automatically.

Note:

This procedure applies to public cache managers only. You cannot reload a system cache manager.

> To reload a cache manager

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the reload icon  for the cache manager that you want to reload.
4. When prompted to confirm that you want to reload the cache manager, click **OK**.

Deleting a Cache Manager

Use the following procedure to delete a cache manager.


When you delete a cache manager, Integration Server deletes all of the caches associated with the cache manager and their contents.

If the cache manager contains distributed caches, those caches are no longer available to this Integration Server. However, the caches are not deleted from the Terracotta Server Array and they remain available to other Integration Servers that are connected to the Terracotta Server Array.

Note:

This procedure applies to public cache managers only. You cannot delete a system cache manager.

> To delete a cache manager

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the delete icon  for the cache manager that you want to delete.
4. When prompted to confirm that you want to delete the cache manager, click **OK**.

Reinitializing a Cache Manager

When you make certain kinds of changes to a cache manager or its caches, you must reinitialize the cache manager to put the changes into effect. You can reinitialize a cache manager in any of the following ways:

- Restarting the Integration Server.

- Starting the cache manager (if it is not already running). For procedures, see [“Starting a Cache Manager” on page 778](#).
- Reloading the cache manager (if it is already running). For procedures, see [“Reloading a Cache Manager” on page 778](#).

Note:

As an alternative to reloading the cache manager, you can stop the cache manager and restart it. These steps are equivalent to doing a reload. For procedures, see [“Shutting Down a Cache Manager” on page 777](#) and [“Starting a Cache Manager” on page 778](#).

Adding a Cache Manager Configuration File to Integration Server

If you have an existing cache manager configuration file (for example, one that you created manually or one that you copied from another Integration Server), you can add it to your Integration Server using the following procedure.

If you are adding a configuration file that you created manually (rather than copying a configuration file from another Integration Server), verify that the file satisfies the following conditions before you perform this procedure:

- The name of the file ends with the extension “.xml”.
- The name of the file does not begin with the characters “SoftwareAG”.
- The <ehcache> element includes the name attribute. The name attribute specifies the name of the cache manager (the name under which the cache manager will be registered in Integration Server). Make sure the name does not begin with the characters “SoftwareAG”. Make sure the name is unique among cache managers on the Integration Server to which you are copying the file.

Example: <ehcache name="MyCacheManager">

- The rejoin attribute is set to “true” if the cache manager is configured to use a Terracotta Server Array. In other words, if the <terracottaConfig> element is present and its url attribute is specified, make sure the rejoin attribute is also specified and that it is set to “true.”

Example: <terracottaConfig url="srv3A:9510,srv3B:9510" rejoin="true">

- All distributed caches that are defined in the configuration file include the <nonstop> element. Integration Server requires all distributed caches to run in nonstop mode.

Example:

```
<cache
name="JLSCache01"
eternal="true"
maxElementsInMemory="5000"
overflowToDisk="false"
copyOnWrite="true"
copyOnRead="true">
<terracotta> <nonstop></nonstop> </terracotta>
```



```
</cache>
```

Note:

If you have created your own configuration file, be aware that Integration Server Administrator does not preserve comments when it updates a configuration file. If your configuration file includes comments, those comments will be discarded the first time someone edits it using Integration Server Administrator.

➤ **To manually add a cache manager to an Integration Server**

1. Copy your configuration file to the *Integration Server_directory* \instances\ *instance_name* \config\Caching directory.
2. Restart Integration Server to initialize the cache manager.

Note:

If the file that you copied to Integration Server in step 1 replaced the configuration file for a cache manager that is already registered on Integration Server, you can reload the cache manager using Integration Server Administrator instead of restarting the Integration Server. (If the cache manager is currently shut down, start it first and then reload it.) For procedures, see [“Reloading a Cache Manager” on page 778](#).

3. Open Integration Server Administrator and go to **Settings > Caching** to verify that Integration Server was able to add your cache manager successfully.

If the cache manager is not listed on the **Settings > Caching** page or if it is not functioning properly, consult the server log or the Ehcache log to see whether errors occurred during the initialization process. For information about viewing the Ehcache log, see [“Logging Ehcache Activity” on page 797](#). If there are problems with the configuration file (for example, if Integration Server could not parse the file or the file contains a duplicate cache manager name), Integration Server records an error message in the server log and writes a full stack trace to the error log.

Manually Editing a Cache Manager Configuration File

If you are familiar with the structure of an Ehcache configuration file, you can use the following procedure to modify the parameters in a cache manager configuration file. You might do this, for example, to add or edit certain cache parameters that are not shown in Integration Server Administrator.

Note:

Generally speaking, you should use the Integration Server Administrator to modify a cache manager or a cache. Edit the configuration file manually *only* when you need to edit parameters that are not exposed in the Integration Server Administrator user interface.

Be aware that when you view a configuration file that was generated by Integration Server Administrator, the file will not contain entries for parameters that use their default settings. Integration Server Administrator only writes required parameters to the configuration file and

parameters whose values are different than their default values. For example, if you create a new cache using Integration Server Administrator and you leave the **Disk Expiry Thread Interval** at its default setting, the Integration Server Administrator does not include the `<DiskExpiryThreadIntervalSeconds>` element in the configuration file. It includes this element only if you change **Disk Expiry Thread Interval** from its default setting.

➤ **To manually edit a cache manager configuration file**

1. Go to the `Integration Server_directory \instances\instance_name\config\Caching` directory and locate the configuration file that you want to edit.
2. As a precaution, make a backup copy of the configuration file before you proceed.

Note:

Because Integration Server attempts to generate cache managers from the XML files it finds in the `Integration Server_directory \instances\instance_name\config\Caching` directory, either store your backup copy in another directory or change its name so that it does not have an ".xml" file extension.

3. Open the configuration file in a text editor and modify it as needed. When editing the file, keep the following points in mind:
 - Integration Server requires all cache managers that connect to a Terracotta Server Array to run with `rejoin="true"` attribute is present in the `<terracottaConfig>` element, do not remove it.
 - Integration Server requires all distributed caches to run in nonstop mode. Never remove the `<nonstop>` option from a distributed cache definition.
 - Integration Server does not preserve comments in its cache manager configuration files. If you add comments to the configuration file while you are editing it, be aware that those comments will be discarded the first time that someone edits the configuration file using Integration Server Administrator.

For information about the parameters in a cache manager configuration file, see Ehcache product documentation for 2.8 at <http://ehcache.org/documentation>.

4. After you save the configuration file, use one of the following techniques to reinitialize the cache manager whose configuration file you edited.
 - Restart Integration Server.
 - Start the cache manager (if it is not already running) and then immediately reload it (to force it to read the updated configuration file from disk). For procedures, see ["Starting a Cache Manager" on page 778](#) and ["Reloading a Cache Manager" on page 778](#).
 - Reload the cache manager (if it is already running). For procedures, see ["Reloading a Cache Manager" on page 778](#).

5. Use Integration Server Administrator to view the cache manager and its caches. If you need procedures for this step, see [“Viewing or Modifying a Cache Manager” on page 776](#) or [“Viewing or Modifying Cache Settings” on page 794](#).

If the cache manager or the cache is not listed in the Integration Server Administrator or if it is not functioning properly, consult the server log or the Ehcache log to see whether errors occurred during the initialization process. For information about viewing the Ehcache log, see [“Logging Ehcache Activity” on page 797](#). If there are problems with the configuration file (for example, if Integration Server cannot parse the file or the file contains a duplicate cache manager name), Integration Server records an error message in the server log and writes a full stack trace to the error log.

6. If you modified any of the cache-wide parameters for a distributed cache (which are the **Maximum Elements in Memory**, **Eternal**, **Time to Live**, **Time to Idle**, **Maximum Entries in Cache**, and **Logging** parameters), perform the following steps to put your changes into effect:
 - a. In Integration Server Administrator, click **Caching** in the **Settings** menu of the Navigation panel.
 - b. Click the cache manager containing the cache that you modified.
 - c. Under **Cache List**, click the cache that you modified.
 - d. Click **Save Changes**.

This step puts the cache-wide settings into effect on this Integration Server and all other Integration Servers that use the distributed cache.

Working with Caches

The procedures in this section describe how to create, edit, clear, disable, enable, and delete a cache.

Creating a Cache

Use the following procedure to create a cache.

Note:

You cannot add a cache to a system cache manager.

You must complete the following before you create a cache:

- The cache manager to which you want to add the cache must already exist. If you need to create the cache manager, see [“Creating a Cache Manager” on page 775](#).
- If you are creating a distributed cache, your Integration Server must have a license to use a Terracotta Server Array. In addition, the cache manager to which you want to add the cache

must be configured to use a Terracotta Server Array. For more information about licenses, see [“Installing, Viewing, and Changing the Terracotta License” on page 756](#).

- If you want to create an off-heap cache (a cache that uses BigMemory), your Integration Server must have a license to use BigMemory. For more information about licenses, see [“Installing, Viewing, and Changing the Terracotta License” on page 756](#).

If you are creating a cache for the first time, review the following sections before you begin.

If you want to create... **See the following topic...**

An on-heap cache [“Configuring an On-Heap Cache” on page 758](#).

An off-heap cache [“Configuring a BigMemory Cache” on page 760](#).

A distributed cache [“Configuring a Distributed Cache” on page 763](#).

Note:

The parameter descriptions in the following procedure indicate whether a parameter is *dynamic* or *non-dynamic*. If the parameter applies to a distributed cache, the description also indicates whether it is *cache-wide* or *client-specific*. For more information about dynamic and non-dynamic parameters, see [“Dynamic vs. Non-Dynamic Cache Parameters” on page 756](#). For more information about cache-wide and client-specific parameters, see [“Cache-Wide and Client-Specific Parameters for a Distributed Cache” on page 767](#).

➤ **To create a cache**

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the cache manager to which you want to add a cache.
4. Click **Add Cache**.
5. Under **Cache Configuration**, specify the following information.

Field	Description
Cache Name	Name for the cache. A valid cache name: <ul style="list-style-type: none"> ■ Must be unique within a cache manager. ■ Must not be null. ■ Must not contain these characters:

Field	Description
	<p data-bbox="673 262 1015 294">? [] / \ = + < > : ; “ , * ^ @</p> <p data-bbox="625 325 1453 388">Note: You cannot modify the name of a cache once the cache is created.</p>
Maximum Elements in Memory	<p data-bbox="625 430 1469 556">Total number of elements that this cache can keep in on-heap memory. A value of 0 sets no limit. However, specifying no limit can affect performance. Therefore, Software AG recommends setting this field to a value greater than 0.</p> <p data-bbox="625 588 1453 735">Note: When you are using off-heap memory (BigMemory), Software AG recommends setting this field to at least 100 elements to minimize performance degradation.</p> <p data-bbox="625 766 1161 798">This parameter is: Dynamic. Cache-wide.</p>
Eternal	<p data-bbox="625 829 1469 924">Optional. When selected, indicates that the elements in this cache will never expire once they are put into the cache. If selected, values in Time to Live and Time to Idle are ignored.</p> <p data-bbox="625 955 1096 987">This check box is cleared by default.</p> <p data-bbox="625 1018 1161 1050">This parameter is: Dynamic. Cache-wide.</p>
Time to Live	<p data-bbox="625 1081 1469 1207">Optional. Maximum amount of time (in seconds) that an element can remain in the cache regardless of whether it is accessed or not. A value of 0 indicates that elements in this cache do not have a time-to-live expiration. The default value is 0.</p> <p data-bbox="625 1239 1218 1270">This field is ignored when Eternal is selected.</p> <p data-bbox="625 1302 1453 1438">Note: If you set both Time to Live and Time to Idle to 0, elements in the cache will never expire. Setting both of these fields to 0 is equivalent to selecting the Eternal check box.</p> <p data-bbox="625 1480 1453 1722">Important: If you need to modify this field after creating the cache as part of a distributed cache, <i>do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, first start the cache manager by following the procedure in “Starting a Cache Manager” on page 778.</p> <p data-bbox="625 1753 1161 1785">This parameter is: Dynamic. Cache-wide.</p>
Time to Idle	<p data-bbox="625 1816 1469 1869">Optional. Maximum amount of time (in seconds) that an element can remain in the cache without being accessed. A value of 0</p>

Field	Description
	<p>indicates that elements in this cache do not have a time-to-idle expiration. The default value is 0.</p> <p>This field is ignored when Eternal is selected.</p> <p>Note: If you set both Time to Live and Time to Idle to 0, elements in the cache will never expire. Setting both of these fields to 0 is equivalent to selecting the Eternal check box.</p> <p>Important: If you need to modify this field after creating the cache as part of a distributed cache, <i>do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, first start the cache manager by following the procedure in “Starting a Cache Manager” on page 778.</p> <p>This parameter is: Dynamic. Cache-wide.</p>
Overflow to Disk	<p>Optional. When selected, indicates that Integration Server writes elements to disk when the memory-based portion of on-heap and off-heap cache is filled.</p> <p>If you select this check box, make sure that the key/value pairs used for elements in the cache are Java serializable. Not doing so would result in an exception while using the cache during the execution of public Integration Server services.</p> <p>This check box applies to local caches only and is cleared by default. It is disabled for distributed caches when the Persist to Disk check box is selected.</p> <p>This parameter is: Non-dynamic. Not applicable to distributed cache.</p>
Persist to Disk	<p>Optional. When selected, indicates that Integration Server maintains a copy of the entire cache on disk so that the state of the cache is preserved when Integration Server is shut down or the cache manager is reinitialized.</p> <p>If you select this check box, make sure that the key/value pairs used for elements in the cache are Java serializable. Not doing so would result in an exception while using the cache during the execution of public Integration Server services.</p> <p>Note: Caches that have Persist to Disk enabled cannot be changed to any other form of persistence once the cache manager is started.</p>

Field	Description
	<p>This check box applies to local caches only and is cleared by default. It is disabled for distributed caches when the Overflow to Disk check box is selected.</p> <p>This parameter is: Non-dynamic. Not applicable to distributed cache.</p>
Maximum Entries Local Disk	<p>Optional. Maximum number of elements that can be maintained in this cache, on-heap, off-heap, and on-disk combined. For example, if you have an on-heap cache of 5000 elements and you want to allow a maximum of 1000 additional elements to overflow to disk, set Maximum Entries Local Disk to 6000.</p> <p>A value of 0 sets no limit. However, specifying no limit can affect performance and can lead to errors if the disk runs out of space. Therefore, Software AG recommends setting this field to a value greater than 0.</p> <p>Note: This field applies to only to local caches and is enabled only when the Overflow to Disk check box is selected.</p> <p>This parameter is: Dynamic. Client-specific.</p>

6. If you are creating a distributed cache, select the **Distributed** check box and then configure the following fields under **Distributed Cache Configuration**:

Note:
You can only select the **Distributed** check box if 1) Your Integration Server has the licenses necessary to create a distributed cache, and 2) The **Terracotta Server Array URLs** field is configured in the cache manager to which you are adding the cache. For more information about licenses, see [“Installing, Viewing, and Changing the Terracotta License” on page 756](#).

Field	Description
Consistency	<p>Model to use to ensure consistency of elements across all caches in the cluster. Select one of the following:</p> <ul style="list-style-type: none"> ■ strong guarantees the consistency of elements across the cluster at all times by immediately synchronizing updates to an element across the cluster. This is the default. ■ eventual guarantees eventual consistency of elements across the cluster. With this setting, updates to an element are not immediately synchronized across the cluster.

Field	Description
Maximum Entries in Cache	<p>Important: Do not change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, follow the procedure in “Modifying Settings for a Distributed Cache” on page 794.</p> <p>This parameter is: Non-dynamic. Client-specific.</p> <p>The maximum size (in number of elements) to which this cache can grow on the Terracotta Server Array. You must specify a value greater than 0.</p> <p>Important: Do not change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, follow the procedure in “Modifying Settings for a Distributed Cache” on page 794.</p> <p>This parameter is: Dynamic. Cache-wide.</p>
Enable High Availability	<p>When selected, indicates that you want to enable nonstop behavior for the distributed cache. Nonstop behavior allows Integration Server to take a prescribed action if an operation on the distributed cache does not complete within a specified timeout period (for example, if the Terracotta Server Array is unresponsive because it is busy or if the network connection is down). For more information about nonstop behavior, see “The Nonstop Behavior of a Distributed Cache” on page 770.</p> <p>Note: Integration Server automatically selects this check box for a distributed cache. You cannot clear this check box.</p>
Timeout	<p>Optional. Amount of time (in milliseconds) that must elapse before Integration Server determines that it cannot reach the Terracotta Server Array and performs the time-out measure specified in Timeout Behavior. The default value is 30000.</p> <p>This parameter is: Dynamic. Client-specific.</p>
Timeout Behavior	<p>Step you want Integration Server to take in order to complete a cache operation when the Terracotta Server Array does not respond within the amount of time specified in Timeout. Select one of the following:</p> <ul style="list-style-type: none"> ■ exception - Returns a NonStopCacheException. This is the default.

Field	Description
	<ul style="list-style-type: none"> <li data-bbox="625 262 1487 367">■ noop - Returns a null for “get” operations. Ignores operations that modify elements in the cache (for example, “put,” “replace,” and “remove” operations). <li data-bbox="625 388 1487 535">■ localReads - Returns elements that reside in the local cache in response to “get” operations. Ignores operations that modify elements in the cache (for example, “put,” “replace,” and “remove” operations). <p data-bbox="625 556 1193 598">This parameter is: Dynamic. Client-specific.</p>
<p data-bbox="284 598 609 682">Immediate Timeout When Disconnected</p>	<p data-bbox="625 598 1487 745">Optional. When selected, indicates that Integration Server should automatically time out all subsequent operations for this cache if Integration Server becomes disconnected from the Terracotta Server Array.</p> <p data-bbox="625 766 1487 987">If you select this check box when Integration Server is disconnected, Integration Server will immediately time out all subsequent operations on the cache instead of waiting for the timeout period specified in Timeout. Integration Server will begin processing cache operations normally again when it successfully rejoins the Terracotta Server Array.</p> <p data-bbox="625 1008 1096 1050">This check box is cleared by default.</p> <p data-bbox="625 1071 1193 1113">This parameter is: Dynamic. Client-specific.</p>
<p data-bbox="284 1123 609 1165">Synchronous Writes</p>	<p data-bbox="625 1123 1487 1438">Optional. When selected, indicates that synchronous writes to the Terracotta Server Array are allowed. Selecting this check box maximizes data integrity by requiring Integration Server to wait for a “transaction received” acknowledgment from a server in the Terracotta Server Array before considering the write operation to have completed. However, enabling synchronous writes can significantly degrade the performance of distributed caches. Conversely, clearing this check box improves performance but may compromise the integrity of the data.</p> <p data-bbox="625 1459 1487 1543">This check box is only available when Consistency is set to strong. It is cleared by default.</p> <p data-bbox="625 1564 1258 1606">This parameter is: Non-dynamic. Client-specific.</p>

7. If you are creating a local cache and you want the cache to overflow to BigMemory, select the **Overflow to Off-heap** check box and then configure the following field under **BigMemory**:

Note:

You can only select the **Overflow to Off-heap** check box if your Integration Server is licensed to use BigMemory. For more information about licenses, see [“Installing, Viewing, and](#)

“[Changing the Terracotta License](#)” on page 756. For more information about things to consider when creating a BigMemory cache, see “[Configuring a BigMemory Cache](#)” on page 760.

Field	Description
Maximum Off-heap	<p>Maximum amount of off-heap memory that this cache can use. Add one of the following suffixes to the amount to indicate the unit of measure:</p> <ul style="list-style-type: none"> ■ k or K for kilobytes ■ m or M for megabytes ■ g or G for gigabytes ■ t or T for terabytes <p>For example, 2g allocates 2 gigabytes to off-heap memory. The minimum amount you can allocate is 1 megabyte. There is no maximum.</p> <p>This field applies to local caches only and is available only if Overflow to Off-heap is selected.</p> <p>This parameter is: Non-dynamic. Not applicable to distributed cache.</p>

8. Configure the following fields under **Cache Configuration Advanced Settings**:

Field	Description
Disk Expiry Thread Interval	<p>Optional. Amount of time (in seconds) that Integration Server waits before checking for and removing expired elements based on the values set for Time to Live and Time to Idle. The default value is 120 seconds.</p> <p>This field applies to local caches only and is enabled only when the Overflow to Disk check box is selected. It is disabled for distributed caches.</p> <p>This parameter is: Non-dynamic. Not applicable to distributed cache.</p>
Disk Spool Buffer Size	<p>Optional. Size (in megabytes) allocated on the disk store for the spool buffer that this cache uses to improve the performance of write operations. The default size is 30 megabytes.</p> <p>Consider increasing the value of this field to improve disk store performance, or decreasing the value if you start to receive OutOfMemory errors.</p>

Field	Description
	<p>This field applies to local caches only and is enabled only when the Overflow to Disk check box is selected. It is disabled for distributed caches.</p> <p>This parameter is: Non-dynamic. Not applicable to distributed cache.</p>
Clear on Flush	<p>Optional. When selected, indicates that memory should be cleared when a “flush” operation is performed on the cache.</p> <p>This check box is selected by default.</p> <p>This parameter is: Non-dynamic. Client-specific.</p>
Copy on Read	<p>Optional. When selected, indicates that a copy of the element is returned when the element is read from the cache. Any changes you make to the element affect only the copy. When you put the copy of the element back into the cache, the changes become visible to other users of the cache.</p> <p>When this check box is cleared, the element is not copied. Any changes you make to the element affect the element directly, through its reference, even if a “put” operation has not been performed to replace the element in the cache.</p> <p>This check box is selected by default.</p> <p>This parameter is: Non-dynamic. Client-specific.</p> <p>The corresponding property is <code>watt.server.serviceResults.copyOnRead</code>.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: Selecting this check box provides a degree of safety against certain logic errors in your application. However, each “get” operation on the cache causes a copy of the element to be created. This can affect performance, especially for large elements. Alternatively, consider structuring your application code so that cache gets are isolated, and references to cached objects are not retained by the application.</p> <p>Important: <i>Do not</i> change this value if the cache manager is shut down. Doing so will cause Integration Server to issue an exception and the cache manager will not start. To change this parameter, follow the procedure in “Modifying Settings for a Distributed Cache” on page 794.</p> </div>
Copy on Write	<p>Optional. When selected, indicates that a copy of the element is placed in the cache during a put operation. Any subsequent changes</p>

Field	Description
	<p>you make to the element affect only the copy. When you put the copy of the element back into the cache, the changes become visible to other users of the cache.</p> <p>When this check box is cleared, put operations pass the element by reference. If you retain a reference to the element after it is put into the cache, any changes you make to the element will also modify the element in the cache, even if you do not do a subsequent put operation.</p> <p>This check box is selected by default.</p> <p>This parameter is: Non-dynamic. Client-specific.</p> <p>The corresponding property is <code>watt.server.serviceResults.copyOnWrite</code>.</p>
<p>Memory Store Eviction Policy</p>	<p>Note: Selecting this check box provides a degree of safety against certain logic errors in your application. However, each “put” operation on the cache causes a copy of the element to be created. This can affect performance, especially for large elements. Alternatively, consider structuring your application code so that cache “put” operations are isolated, and references to cached objects are not retained by the application.</p> <p>Optional. Policy used to remove elements from the memory store when the value in Maximum Elements in Memory is reached. Select one of the following:</p> <ul style="list-style-type: none"> ■ LRU (Least Recently Used). This is the default. ■ LFU (Least Frequently Used) ■ FIFO (First In First Out). This setting is not available for distributed caches. <p>This parameter is: Dynamic. Client-specific.</p>
<p>Logging</p>	<p>Optional. When selected, indicates that basic distributed-map logging messages are saved to the Terracotta logs.</p> <p>This check box is cleared by default.</p> <p>Note: When this check box is selected for a distributed cache, logging events are recorded on the Terracotta Server Array. For more information about logging, see “Logging Ehcache Activity” on page 797.</p>

Field	Description
	This parameter is: Non-dynamic. Client-specific.

9. To make the cache searchable, select the **Searchable** check box and configure the following field under **Search Configuration**:

Field	Description
Allow Automatic Indexing	Optional. When you select either Key , Value , or both, Integration Server automatically indexes the searchable keys and values. If you are not planning to include the keys and values in your query, do not enable automatic indexing.

Note:

If you select automatic indexing, you cannot have a mix of search types for keys or values.

This parameter is: Non-dynamic. Cache-wide.

10. Under **Search Attributes**, define one or more attributes that will be used in searches by configuring the following fields:

Field	Description
Attribute	Name of the attribute.
Extract Method	<p>The method to use to extract the search attributes from keys or values. Select one of the following:</p> <ul style="list-style-type: none"> ■ Expression to specify expressions based on which attributes are extracted from keys or values. Enter the expression in the Class/Expression field. For example, <code>key.name</code>, <code>value.age</code>. ■ Class to provide the extractor class and, optionally, the properties for the extractor class. <p>Enter the extractor class in the Class/Expression field.</p> <p>In the Properties field, specify the properties as key/value pairs separated by commas (,).</p> <p>The extractor must be an implementation of <code>net.sf.ehcache.search.attribute.AttributeExtractor</code>. For more information about attribute extractors, see “Defining Attributes” on page 772 and Ehcache product documentation.</p>
Add/Remove	Click + to add or × to delete search attributes to use in searches.

11. Click **Save Changes**. Integration Server creates the cache and places it in the disabled state.
12. When you are ready to put the cache into use, enable it using the procedure in [“Enabling a Cache” on page 795](#).

Viewing or Modifying Cache Settings

Use the following procedure to view cache settings or modify the cache-wide settings for an existing cache. You can modify client-specific settings by manually editing the cache manager configuration file. For more information, see [“Manually Editing a Cache Manager Configuration File” on page 781](#).

Note:

This procedure describes how to modify a cache that is not distributed. For information about modifying cache settings for a distributed cache, see [“Modifying Settings for a Distributed Cache” on page 794](#).

Note:

Be aware that many of the parameters associated with a system cache are system-defined and cannot be edited in Integration Server Administrator if Integration Server is not running in quiesce mode. When you edit a system cache, you can see all of its parameters, but you can edit only certain settings. If you are editing a system cache for another webMethods product, such as Trading Networks, consult the product documentation for guidelines.

➤ To view or modify cache settings

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Click the cache manager containing the cache that you want to view or modify.
4. Under **Cache List**, click the cache you want to view or modify.
5. View cache settings or modify the cache-wide settings as necessary.

For more information about the caches settings, see [“Creating a Cache” on page 783](#).

6. If you have made changes to the cache settings, click **Save Changes** to save them.

Modifying Settings for a Distributed Cache

Use the following procedure to modify settings for distributed caches.

➤ To modify settings for a distributed cache

1. Shut down the cache manager on all clients. For more information, see “[Shutting Down a Cache Manager](#)” on page 777.
2. Use the Terracotta Management Console to delete the offline data. For more information about the Terracotta Management Console, see Ehcache product documentation for 2.8 at <http://ehcache.org/documentation>.
3. Modify the necessary settings in the cache manager configuration file. For more information, see “[Manually Editing a Cache Manager Configuration File](#)” on page 781.
4. Restart the cache manager. For more information, see “[Starting a Cache Manager](#)” on page 778.

Disabling a Cache

You can disable a cache to troubleshoot a load or performance issue, for example. Any “get” operations performed against a disabled cache return null, and any “put” operations are discarded. Existing cache elements remain in the cache and can be used again when the cache is enabled (assuming the cache manager has not been initialized in the meantime).

If the cache is distributed, the cache is disabled only on the Integration Server where the disable operation is performed. The cache still resides on the Terracotta Server Array and remains accessible to other Integration Servers connected to the Terracotta Server Array.

Note:

You can only disable a cache that belongs to a public cache manager.

➤ To disable a cache

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the cache manager associated with the cache you want to disable.
4. Under **Cache List**, click the **Yes** link in the **Enabled** column next to the cache you want to disable.
5. When prompted to confirm that you want to disable the cache, click **OK**.

Enabling a Cache

A cache becomes enabled automatically when the cache manager it belongs to is started or reloaded. “Put” and “get” operations can be performed against an enabled cache. When you enable a disabled cache, all the elements stored previously in the cache are again available (assuming the cache manager has not been initialized in the meantime).

Note:

To enable a disabled cache, the cache manager must be running. You cannot enable a cache while its cache manager is shut down.

Note:

You can only enable a cache that belongs to a public cache manager.

> To enable a cache

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the cache manager associated with the cache you want to enable.
4. Under **Cache List**, click the **No** link in the **Enabled** column next to the cache you want to enable.
5. When prompted to confirm that you want to enable the cache, click **OK**.

Clearing a Cache


When you clear a cache, you remove all elements from that cache.

If the cache is distributed, this procedure removes all elements from that cache on the Terracotta Server Array.

Note:

You can only clear a cache that belongs to a public cache manager.

> To clear a cache

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the cache manager associated with the cache you want to clear.
4. Under **Cache List**, click the clear icon  next to the cache that you want to clear.
5. When prompted to confirm that you want to clear the cache, click **OK**.

Note:

If the clear operation encounters any previously acquired locks on elements, the operation will wait until all the locks are released. As a result, if the locks are not released, the clear operation may wait indefinitely to complete.

Deleting a Cache


Deleting a cache removes the cache from the cache manager. If the cache is local, the existing elements in the cache are discarded.

If the cache is distributed, the cache is no longer available to the Integration Server from which it is deleted. The cache remains available on other Integration Servers connected to the Terracotta Server Array.

Note:

You can only delete a cache that belongs to a public cache manager.

> To delete a cache

1. Open Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Caching**.
3. Under **Public Cache Managers**, click the cache manager associated with the cache that you want to delete.
4. Under **Cache List**, click the delete icon  next to the cache that you want to delete.
5. When prompted to confirm that you want to delete the cache, click **OK**.

Logging Ehcache Activity

Ehcache uses Simple Logging Facade for Java (SLF4J) and its log4J facility to log information about Ehcache activities on Integration Server. Ehcache also creates additional log files when cache managers connect to the Terracotta Server Array.

Logging Ehcache Caching Services

Ehcache logs its messages to the console. Integration Server redirects the Ehcache log messages to the ehcache.log file located in the *Integration Server_directory* \instances*instance_name*\logs directory.

Ehcache logging has a default configuration, but you can change it by modifying the `tc.dev.log4j.properties` file. This file is located in the *Integration Server_directory* directory. The logging level is set to Warn. When the logging level is set to Warn, it includes all the Fatal, Error, and Warn messages that are logged by Ehcache.

Note:

For details about log4J, refer to Apache logging services at <http://logging.apache.org/log4j/1.2/index.html>.

Logging Cache Manager Activity in the Terracotta Server Array

Ehcache creates additional log files on the server containing log data you can provide to Software AG Global Support for troubleshooting assistance. Ehcache creates these log files in the following situations:

- When an Integration Server in a cluster starts up and connects to the Terracotta Server Array.
- When a public cache manager containing distributed caches reloads or starts up.

You specify where you want Ehcache to put the log files by setting the `watt.server.cachemanager.logsDirectory` property on Integration Server. The default value for this property is `Integration Server_directory \instances\instance_name\logs\tc-client-logs`. For information about how to edit this property, see [“Working with Extended Configuration Settings” on page 127](#).

Note:

During start up Integration Server copies the value of the `watt.server.cachemanager.logsDirectory` property into the `com.softwareag.tc.client.logs.directory` property. When you configured the `tc-config.xml` file for the Terracotta Server Array, you defined this property for the `<client><logs>` setting. For information about configuring the `tc-config.xml` file, see [“Configuring tc-config.xml on the Terracotta Server Array” on page 765](#).

41 Configuring the Enhanced XML Parser

■ What Is the Enhanced XML Parser?	800
■ How Does the Enhanced XML Parser Work?	800
■ When Is the Enhanced XML Parser Used?	802
■ Configuring the Enhanced XML Parser	803
■ Setting the Partition Size	806
■ Viewing Peak Usage Statistics	807

What Is the Enhanced XML Parser?

The enhanced XML parser is an XML parser within Integration Server that leverages Ehcache during the parsing process. Depending on memory availability, the enhanced XML parser will use Ehcache to move already parsed portions of the document to a local disk store or off-heap cache (BigMemory). By migrating data, the enhanced XML parser frees up heap space for other documents or other system processes. The ability to move data to a local disk store or BigMemory enables the parser to parse multiple XML documents, including multiple large XML documents, without monopolizing the JVM heap.

The enhanced XML parser produces a DOM tree, specifically a tree of `org.w3c.dom.Node` objects, that is read-only. The DOM tree is a special representation of an XML document that can be consumed by any program that uses the standard DOM API. Integration Server built-in services that took a node produced by the legacy XML parser as input have been updated to take a DOM node as well. Integration Server also includes services for using the enhanced XML parser to create a DOM node from an XML document.

In contrast, the legacy XML parser parses an XML (or HTML) document by creating a collection of nodes that remain in the heap during the entire parsing process. The legacy parser produces a tree of `com.wm.lang.xml.Node` objects, which is a tree of nodes that is similar in structure to a DOM object.

Because the legacy XML parser maintains all of the nodes on the heap while parsing, large documents or multiple documents being parsed at once result in a large number of objects on the heap. This can degrade performance during garbage collection. Additionally, if the legacy XML parser exceeds the amount of available heap while parsing, the Java virtual machine (JVM) throws an `OutOfMemoryError`. To handle large documents with the legacy XML parser, you can increase the heap size or use services like the `queryXMLNode` service or the `nodeIterator` services in `pub.xml` to limit the portions of the node maintained in the heap. The enhanced XML parser handles large documents, multiple documents, and multiple large documents more efficiently by providing memory management without the use of other services.

Note:

While the legacy XML parser can create a node from an XML document or an HTML document, the enhanced XML parser parses XML documents only.

For more information about Ehcache and BigMemory, see [“Configuring Ehcache on Integration Server” on page 749](#).

How Does the Enhanced XML Parser Work?

The enhanced XML parser parses a document by serializing it into partitions that reside on the JVM heap. As the number of partitions and the amount of heap space required to parse the document increases, the parser uses Ehcache to move partitions to a local disk store or off-heap cache (BigMemory). Integration Server retrieves the partitions as needed to fulfill subsequent processing requests.

The enhanced xml parser uses heap allocation limits to determine when to migrate partitions to a local disk store or BigMemory. The heap allocation limits enable the enhanced XML parser to

throttle its use of heap. This reduces the chances of unstable behavior when the Integration Server operates at peak capacity. You can set limits for:

- The total heap space used by any single parsed document.
- The total heap space used by all parsed documents combined.
- The total amount of BigMemory used by all parsed documents

Following is an overview of how the enhanced XML parser parses an XML document:

1. The enhanced XML parser receives a request to parse a document. To begin parsing a document, the enhanced XML parser creates six partitions of the same size in the heap. The size of the partitions is determined by the *partitionSize* value passed to the service or the default partition size configured for the enhanced XML parser. For more information about the partition size, see [“Setting the Partition Size” on page 806](#).

However, if there is not sufficient heap space to create six partitions, the enhanced XML parser delays parsing the document until sufficient heap space becomes available. There might not be sufficient heap space if the amount of heap space currently used for parsed documents is at or near the limit specified by the **Maximum heap allocation for all documents combined** field.

2. The enhanced XML parser serializes the XML document into the partitions on the heap, creating more partitions of the same size as needed.
3. During parsing, if the total heap space required to parse the document meets the limit specified by the **Maximum heap allocation for any single document** field or the limit specified by **Maximum heap allocation for all documents combined**, one of the following occurs:
 - If caching is not enabled, meaning that the parser does not use Ehcache to help manage memory, the JVM may throw an `OutOfMemoryError`. The enhanced XML parser will not finish parsing the document, and the service from which the parser was called will fail.
 - If caching is enabled, but BigMemory is not, the enhanced XML parser uses Ehcache to move some partitions from the heap to a local disk store.
 - If caching is enabled and BigMemory is enabled, the enhanced XML parser uses Ehcache to move some partitions from the heap to BigMemory. If the amount of BigMemory used by the enhanced XML parser exceeds the limit set by the **Maximum BigMemory allocation** field, the enhanced XML parser throws an `UnexpectedCachingException`.

In rare circumstances, the enhanced XML parser might throw the following exceptions:

- If the aggregate attribute value length for any single XML element exceeds 65535 characters, the enhanced XML parser throws an `ImplementationLimitCachingException`.
- If the partition size is relatively small and an individual XML element contains a very large number of attributes or the aggregate length of the attribute values for an element is very long, Integration Server might throw a `PartitionSizeCachingException`. To resolve this issue, increase the partition size. For more information about the partition size, see [“Setting the Partition Size” on page 806](#).

When Is the Enhanced XML Parser Used?

Integration Server uses the enhanced XML parser to parse an XML document only when the enhanced parser is specifically engaged. The enhanced XML parser can be engaged in one of the following three ways:

- By invoking `pub.xml:loadEnhancedXMLNode`.
- By invoking `pub.xml:xmlStringToEnhancedXMLNode`.
- By an HTTP/S POST request with a Content-Type of `text/xml` or `application/xml` that posts a request to a target service and one of the following is true:
 - The request specifies `xmlFormat=enhanced`.
 - The request does not specify an `xmlFormat` value but the target service has a **Default xmlFormat** property value of `enhanced`.
 - The request does not specify an `xmlFormat` value, the **Default xmlFormat** property on the target service is blank, and `watt.server.http.xmlFormat` is set to `enhanced`.

For more information about `xmlFormat` in HTTP requests and the **Default xmlFormat** property, see *webMethods Service Development Help*.

Any services or applications that used the legacy XML parser will continue to do so. For the enhanced XML parser to be used in an existing service or application, you must change the service or application to call the enhanced XML parser using one of the methods listed above.

Note:

For existing services or applications, Software AG does not recommend engaging the enhanced XML parser by changing the `watt.server.http.xmlFormat` parameter to `enhanced`. This is a global parameter that affects the entire Integration Server. Changing the default `xmlFormat` for all of Integration Server instead of changing the `xmlFormat` on service or application basis might cause existing services and applications to fail.

What Services Consume an Enhanced Node?

Most existing `WmPublic` built-in services that took a node produced by the legacy XML parser as input (specifically a node of type `com.lang.wm.xml.Node`) can also take a node produced by the enhanced XML parser. This includes the following services:

- `pub.xml:xmlNodeToDocument`
- `pub.xml:queryXMLNode`
- `pub.xml:getXMLNodeIterator`
- `pub.xml:freeXMLNode`

The exception is the `pub.schema.validate` service. This service cannot be used to validate a node produced by the enhanced XML parser.

Configuring the Enhanced XML Parser

When you configure the enhanced XML parser, you indicate the memory usage limits for the parser and whether or not partitions will overflow to a local disk store or BigMemory when the usage limits are met.

Keep the following information in mind when configuring the enhanced XML parser.

- To take advantage of the memory management provided by the enhanced XML parser, you must enable caching. When caching is enabled, the enhanced XML parser uses Ehcache to limit the amount of heap used for parsing. When caching is not enabled, the enhanced XML parser behaves in much the same way as the legacy XML parser.
- If only caching is enabled, the enhanced XML parser uses Ehcache to move partitions to a local disk store when heap allocation limits are met.
- If caching is enabled and BigMemory is enabled, the enhanced XML parser uses Ehcache to move partitions to BigMemory when heap allocation limits are met.
- To use BigMemory with the enhanced XML parser, caching must be enabled. Additionally, the Integration Server must be equipped with a license to use BigMemory. To verify that the Integration Server has a BigMemory license and to check how much BigMemory the license authorizes, see [“Installing, Viewing, and Changing the Terracotta License” on page 756](#).

➤ To configure the enhanced XML parser

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Enhanced XML Parsing**.
3. If you want the enhanced XML parser to use Ehcache for memory management, under **Caching**, make sure the **Enabled** property is set to **Yes**. If **Enabled** is not set to **Yes**, click **No** to change the value. Integration Server Administrator prompts you to confirm enabling the Caching feature. Click **OK**.
4. If you want the enhanced XML parser to make use of BigMemory (off-heap cache), under **BigMemory Caching**, make sure the **Enabled** property is set to **Yes**. If **Enabled** is not set to **Yes**, click **No** to change the value. Integration Server Administrator prompts you to confirm enabling the BigMemory caching feature. Click **OK**.

Note:

To use BigMemory with the enhanced XML parser, you must enable caching as well.

5. At the top of the **Settings > Enhanced XML Parsing** page, click **Edit Enhanced XML Parsing Settings**.

6. Under **Enhanced Parser Settings**, in the **Default partition size** field, specify the default size of the partitions on the heap where the enhanced XML parser stores parsed document information. Add one of the following suffixes to indicate a unit of measure:

- k or K for kilobytes
- m or M for megabytes

The default is 20k.

Note: Integration Server uses the default partition size when parsing XML documents received by services that specify an `xmlFormat` of “enhanced”. For more information about `xmlFormat`, see *webMethods Service Development Help*. The `pub.xml:loadEnhancedXMLNode` service and the `pub.xml:xmlStringToEnhancedXMLNode` service use the default partition size if the services do not specify a value for the `partitionSize` input parameter.

7. Under **Caching**, specify the following information:

For this field	Specify
Maximum heap allocation for all documents combined	<p>Maximum amount of on-heap space that the enhanced XML parser can allocate to process documents concurrently. Add one of the following suffixes to the amount to indicate a unit of measure:</p> <ul style="list-style-type: none"> ■ k or K for kilobytes ■ m or M for megabytes ■ g or G for gigabytes ■ % to indicate a percentage of the heap space

The default is 40% of the heap space available to the JVM.

Note:

If you do not specify a suffix, Integration Server uses bytes as the unit of measure.

When the total heap used by parsed documents exceeds the **Maximum heap allocation for all documents combined** value, the enhanced XML parser will not begin parsing more documents until heap space becomes available. Additionally, the enhanced XML parser moves some partitions to BigMemory (if BigMemory caching is enabled) or a local disk store (if BigMemory caching is not enabled).

Important:

Do not set **Maximum heap allocation for all documents combined** to be 100% of the heap. A value of 100% indicates that there is no limit to the amount of heap the enhanced XML parser can use. If enough documents are parsed simultaneously, the JVM may throw an `OutOfMemoryError`.

For this field	Specify
Maximum heap allocation for any single document	<p data-bbox="576 262 1453 367">Maximum amount of on-heap space that the enhanced XML parser can allocate to process a single document. Add one of the following suffixes to the amount to indicate a unit of measure:</p> <ul data-bbox="576 388 1469 598" style="list-style-type: none"> ■ k or K for kilobytes ■ m or M for megabytes ■ g or G for gigabytes ■ % to indicate a percentage of the heap space available to the JVM <p data-bbox="576 619 1055 661">The default is 20% of the heap space.</p> <div data-bbox="576 672 1453 808" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="576 682 657 714">Note:</p> <p data-bbox="576 724 1421 787">If you do not specify a suffix, Integration Server uses bytes as the unit of measure.</p> </div> <p data-bbox="576 819 1453 997">When the total heap used by a single parsed document exceeds the Maximum heap allocation for any single document value, the enhanced XML parser moves some partitions to BigMemory (if BigMemory caching is enabled) or a local disk store (if BigMemory caching is not enabled).</p> <div data-bbox="576 1008 1453 1247" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="576 1018 730 1060">Important:</p> <p data-bbox="576 1060 1453 1239">Do not set Maximum heap allocation for any single document to be 100% of the heap. A value of 100% indicates that there is no limit to the amount of heap the enhanced XML parser can use. If enough documents are parsed simultaneously, the JVM may throw an <code>OutOfMemoryError</code>.</p> </div>

8. Under **BigMemory Caching**, in the **Maximum BigMemory allocation** field, specify the maximum amount of **BigMemory** that can be allocated to parsing documents with the enhanced XML parser.

- k or K for kilobytes
- m or M for megabytes
- g or G for gigabytes

The default is 200m.

Note:

If you do not specify a suffix, Integration Server uses bytes as the unit of measure.

When the partitions in BigMemory exceed the limit set by the **Maximum BigMemory allocation** value, the enhanced XML parser throws an `UnexpectedCachingException`.

9. Click **Save Changes**.

Changes do not take effect immediately. Changes will take effect when there are no references to any of the parsed documents produced by the enhanced xml parser.

Setting the Partition Size

When the enhanced XML parser parses a document, it places the content into fixed-size partitions. While the parser determines the size of the partitions, you can provide guidance to the parser by specifying the partition size. You can only specify partition size when using the `pub.xml:loadEnhancedXMLNode` service or the `pub.xml.xmlStringtoEnhancedMLNode` service. Both services have an input parameter named *partitionSize*. Keep the following points in mind when setting a partition size:

- The *partitionSize* is a hint for the enhanced XML parser so that it can estimate the amount of heap space needed to parse the document. Often, it not possible to determine the size of an inbound XML document before parsing.
- As a general rule, Software AG recommends a *partitionSize* that is 1/2 the size of the unparsed XML document.
 - A *partitionSize* that is considerably larger than 1/2 the size of the unparsed XML document causes the enhanced XML parser to consume more heap space than necessary but might also improve throughput. However, this can impact the overall performance of Integration Server.
 - A *partitionSize* that is considerably smaller than 1/2 the size of the unparsed XML document causes the enhanced XML parser to create a large number of partitions to parse the document. While this might use less heap space, it may reduce the throughput of the parser.
 - A *partitionSize* that is three times smaller or three times larger than 1/2 the size of the unparsed XML document will likely have little impact on the performance.
- At run time, the enhanced XML parser overrides a *partitionSize* that consumes all of the available heap space.
- At run time, if the *partitionSize* results in an initial heap allocation that exceeds the single document limit set in the **Maximum heap allocation for any single document** field the limit for all documents set in the **Maximum heap allocation for all documents combined** field, the enhanced XML parser reduces the partition size automatically.
- If you do not specify *partitionSize*, the enhanced XML parser uses the default specified in the **Default partition size** field on the **Settings > Enhanced XML Parsing** page in Integration Server Administrator.

When the enhanced XML parser is called via an HTTP POST request, the enhanced XML parser uses the default partition size specified by the **Default partition size** field on the **Settings > Enhanced XML Parsing** page in Integration Server Administrator.

Viewing Peak Usage Statistics

Integration Server tracks peak memory usage statistics for the enhanced XML parser. The statistics cover the time period from when the Integration Server was last started to the current time. Integration Server resets the statistics during start up.

➤ **To view usage statistics for the enhanced XML parser**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Enhanced XML Parsing**.

Under Peak Usage Statistics, Integration Server Administrator displays the following information:

Field	Description
Peak heap allocation for all documents combined	The maximum amount of heap used by the enhanced XML parser for parsing multiple documents at one time.
Peak heap allocation for any single document	Maximum amount of heap used by the enhanced XML parser for parsing one document.
Peak BigMemory use	Maximum amount of BigMemory used by the enhanced XML parser at one time.

42 Configuring WebSockets

■ Overview	810
■ How WebSocket Works	810
■ Setting Up Integration Server for WebSockets	811
■ Configuring a WebSocket Ports	811
■ Configuring a WebSocketSecure Ports	814
■ Viewing WebSocket Server Endpoints for a WebSocket Port	817
■ WebSocket Server Sessions Screen	818
■ Limitations When Using WebSocket Protocol in Integration Server	818

Overview

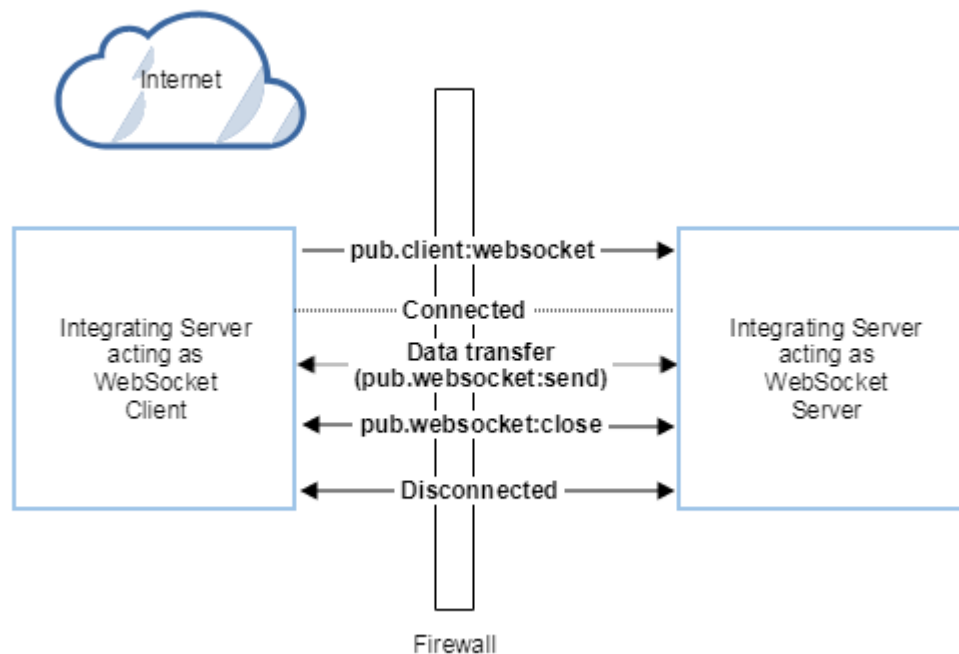
The WebSocket protocol provides for simultaneous two-way communication between a client and server/endpoint over a single TCP connection. Integration Server can act as a WebSocket client endpoint and a WebSocket server endpoint.

How WebSocket Works

In a WebSocket configuration, Integration Server can act as a WebSocket server or a WebSocket client. When Integration Server acts as a WebSocket server, it hosts a server endpoint that you create using Designer. The server endpoint captures the properties required for callback services for the life cycle of a WebSocket. The endpoint is identified by a path within a server.

When Integration Server acts as a WebSocket client it creates a WebSocket client endpoint. The client endpoint captures information related to connection and the respective callback services thereof. It also connects to the WebSocket server identified by a URI.

The following diagram illustrates how a request is handled in a WebSocket



configuration:

- (1). To establish a WebSocket session between a client and a Integration Server, the client needs to send an HTTP Upgrade request. This is done using `pub.client:websocketservice`.
- (2). If Integration Server is capable and willing to upgrade the session, it sends a HTTP 101 response to the requesting client. At this point the handshake is considered successful and the connection between the server and the client is upgraded to the WebSocket protocol. This is done using a capable WebSocket port.

(3). Data can now flow bidirectional between the Integration Server and the client over the WebSocket session. This is done using `pub.websocket:send` service.

(4). Any client or server in the data exchange can request the WebSocket connection be closed by sending a close request. This is done using `pub.websocket:close` service.

Setting Up Integration Server for WebSockets

This section describes the high-level steps for setting up the WebSocket application. The following checklist summarizes these steps:

Done?	Task	Notes
	Configure the WebSocket port.	<p>You can use WebSocket or WebSocketSecure port options depending on your needs. For instructions, see “Configuring a WebSocket Ports” on page 811.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If you plan to use a WebSocketSecure port, you must store a server certificate, a server private key, and a CA certificate on this server. For instructions, see “Configuring Integration Server for Secure Communication” on page 457.</p> </div>
	Configure the WebSocket server endpoint. (Optional) If Integration Server is acting as a Websocket server.	For information about this task, see <i>webMethods Service Development Help</i>
	Configure the WebSocket client endpoint. (Optional) If Integration Server is acting as a Websocket client.	For information about this task, see <i>webMethods Service Development Help</i>

Configuring a WebSocket Ports

The Integration Server that acts as a WebSocket server uses a port to listen for requests from clients through which it maintains its connection.

To add a WebSocket port, complete the following steps.

➤ To add a WebSocket port

1. Open Integration Server Administrator if it is not already open.

- In the **Security** menu of the Navigation panel, click **Ports**.
- Click **Add Port**.
- In the **Add Port** area of the screen, select **webMethods/WebSocket**.
- Click **Submit**. Integration Server Administrator displays a screen requesting information about the port.
- On the **Edit WebSocket Port Configuration** screen, under **webMethods/WebSocket** port, enter the following information:

Parameter	Specify
Enable	Indicate whether to enable (Yes) or disable (No) this WebSocket listener.
Port	The number you want to use for the port. Select a number that is not already in use on this host machine. Important: If you are running multiple Integration Servers on the same host machine, make sure the port numbers used on each server are unique.
Alias	An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a-z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description	A description of the port.
Package Name	The package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port. If you replicate this package, the Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.
Bind Address (optional)	IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.

Parameter	Specify
Backlog	The number of requests that can remain in the queue for an enabled port before Integration Server begins rejecting requests. The default is 6000. The maximum value is 65535.
	<p>Note: This parameter does not apply to disabled ports. Integration Server refuses requests sent to disabled ports.</p>

7. Under **Threadpool Configuration**, enter the following information:

Note: Integration Server uses the private thread pool for requests coming into this port. With this thread option, requests coming into this port will not have to compete with other server functions of the thread.

Parameter	Specify
Threadpool Min	The minimum number of threads Integration Server maintains in this thread pool. The default is 10.
Threadpool Max	The maximum number of threads Integration Server maintains in this thread pool. The default is 200.
Threadpool Priority	Priority with which the Java treats threads from this thread pool. The larger the number, the higher the priority. The default is 5.
	<p>Important: Use this setting with extreme care because it will affect server performance and throughput.</p> <p>When you view the port's details, Integration Server reports the total number of private threadpool threads currently in use for the port.</p>

8. Under **WebSocket Policy**, enter the following information:

Parameter	Specify
Idle Timeout(ms)	When to close the WebSocket client connection if the server has not received a request from the client within this timeout value (300000 milliseconds).

9. Under **Security Configuration**, enter the following information:

Parameter	Specify
Client Authentication	The type of client authentication you want Integration Server to perform for requests that arrive on this WebSocket port. Integration Server uses user name and password.

10. Click **Save Changes**.
11. On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

12. On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Configuring a WebSocketSecure Ports

The WebSocketSecure port enables Integration Server to secure the connections. By default, the WebSocket listener uses the certificates for the default Integration Server SSL key. However, you can configure the listener to use its own private key residing in an Integration Server keystore (file- or SmartCard/HSM-based). For more information, see [“Configuring Server-Side SSL Settings” on page 463](#).

➤ To add a WebSocketSecure port

1. Open Integration Server Administrator if it is not already open.
2. In the **Security** menu of the Navigation panel, click **Ports**.
3. Click **Add Port**.
4. In the **Add Port** area of the screen, select **webMethods/WebSocketSecure**.
5. Click **Submit**. Integration Server Administrator displays a screen requesting information about the port.
6. Under **WebSocket Listener Configuration**, enter the following information:

Parameter	Specify
Enable	Indicate whether to enable (Yes) or disable (No) this WebSocketSecure listener.

Parameter	Specify
Port	<p>The number you want to use for the port. Select a number that is not already in use on this host machine.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: If you are running multiple Integration Servers on the same host machine, make sure the port numbers used on each server are unique.</p> </div>
Alias	<p>An alias for the port that is unique for this Integration Server. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p>
Description	<p>A description of the port.</p>
Package name	<p>Package associated with this port. When you enable the package, the server enables the port. When you disable the package, the server disables the port.</p> <p>If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. This feature is useful if you create an application that expects input on a specific port. The application will continue to work after it is replicated to another server.</p>
Bind Address (optional)	<p>IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.</p>
Backlog	<p>The number of requests that can remain in the queue for an enabled port before Integration Server begins rejecting requests. The default is 6000. The maximum value is 65535.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This parameter does not apply to disabled ports. Integration Server refuses requests sent to disabled ports.</p> </div>

7. Under **Threadpool Configuration**, enter the following information:

Parameter	Specify
Threadpool Min	<p>The minimum number of threads Integration Server maintains in this thread pool. The default is 10.</p>
Threadpool Max	<p>The maximum number of threads Integration Server maintains in this thread pool. The default is 200.</p>

Parameter	Specify
Threadpool Priority	Priority with which the Java treats threads from this thread pool. The larger the number, the higher the priority. The default is 5.

The default is 5.

Important:

Use this setting with extreme care because it will affect server performance and throughput.

8. Under **WebSocket Policy**, enter the following information:

Parameter	Specify
Idle Timeout(ms)	When to close the WebSocket client connection if the server has not received a request from the client within this timeout value (300000 milliseconds).

9. Under **Security Configuration**, enter the following information:

Parameter	Specify
Client Authentication	The type of client authentication you want Integration Server to perform for requests that arrive on this WebSocket port. Integration Server uses username/password.

10. Under **Listener Specific Credentials**, enter the following information:

Note:

Use these settings only if you want to use a different set of credentials from the ones specified on the Certificates Screen.

Parameter	Specify
Keystore Alias	<p>Optional. A user-specified, text identifier for an Integration Server keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p> <p>For more information, see “Creating Keystore Aliases” on page 481.</p>

Parameter	Specify
Key Alias	Optional. The alias for the private key, which must be stored in the keystore specified by the above keystore alias.

- Click **Save Changes**.
- On the **Ports** screen, click **Edit** to change the Access Mode if necessary. You may **Set Access Mode to Allow by Default** or **Reset to default access settings**.

Note:

If the port is of a WebSocket or WebSocketSecure, then the WebSocket endpoints will be displayed.

For more information about setting access mode for a port and controlling IP access for a port, see [“Controlling Access to Resources by Port” on page 487](#)

- On the **Ports** screen, also check the list of ports to ensure that the status in the **Enabled** column is **Yes**. If it is not, click **No** to enable the port.

Viewing WebSocket Server Endpoints for a WebSocket Port

You can view a list of the WebSocket server endpoints for a WebSocket port.

➤ To view the connections to a WebSocket port

- Open the Integration Server Administrator.
- In the Navigation panel of the screen, on the **Security** menu, click **Ports**.
- On the Security > Ports screen, click the port number of a WebSocket port.
- On the Security > Ports > View WebSocket Port Details screen, click **WebSocket Server Endpoints**.

Integration Server Administrator displays the Security > Ports > WebSocket Server Endpoints screen. The **WebSocket - Registered Server Endpoints** table lists the WebSocket server endpoints on that port. For each connection Integration Server Administrator displays the following information:

Field	Description
WebSocket URI	Indicates the server endpoint that the WebSocket client uses to connect.

Field	Description
Active Sessions	Indicates the number of active sessions in the server. Click the current number of sessions to display the WebSocket Server Sessions screen .

Note:

If there are no active sessions, this displays **No Active Sessions found for WebSocket port <portNumber>**.

WebSocket Server Sessions Screen

You can use the WebSocket Server Sessions Screen to view the current active WebSocket server sessions. Clicking on the current number of sessions displays the following information:

Field	Description
Session ID	Displays the value of the hosting session ID.
Local Address	Displays the local address and port of the WebSocket server for that particular session.
Remote Address	Displays the remote address and port of the client for that particular session.
Protocol Version	Displays the WebSocket protocol version. Integration Server supports only WebSocket protocol version 13.
Idle Timeout	Indicates the length of time in milliseconds, the WebSocket can keep an unused connection open.
Secure	Indicates if the WebSocket session is secure. Yes indicates the session is secure. No indicates the session is not secured.

Use the following options to manage WebSocket server sessions for the WebSocket port.

Option	Description
Close	Close the selected WebSocket session.
Disconnect	Disconnects the selected WebSocket session.

Limitations When Using WebSocket Protocol in Integration Server

Integration Server WebSocket module does not support the following with Websockets:

- Authentication with JWT (JSON Web Token)
- Reverse proxy and forward proxy
- Async send support (only blocking send will be supported)
- Partial message support with WebSockets
- Ping/Pong frame
- Reverse invoke

43 Locking Administration and Best Practices

■ Introduction	822
■ Choosing Local Server Locking or VCS Integration Locking	822
■ Disabling and Re-enabling Locking	822
■ Best Practices	824

Introduction

This chapter contains information intended for the server administrator and users who regularly replicate and publish packages as part of the production process.

Choosing Local Server Locking or VCS Integration Locking

You can configure the webMethods Integration Server to support either of the following two forms of locking:

- Local locking, applied within the Integration Server file system.
- Locking resulting from integration with a third-party version control system (VCS) repository; in this case, elements are locked and unlocked as you check them out of and in to the VCS repository.

The locking administration tasks described in this chapter refer to local locking on the Integration Server. If your server environment does not make use of a version control system, configure the Integration Server for local locking as described in the following sections.

You can use the local service development feature in Designer to work with a third party version control system. For more information about local service development, see the *webMethods Service Development Help*

Disabling and Re-enabling Locking

There may be times in which you do not want to implement locking on the Integration Server. If you are a server administrator, you can disable and re-enable locking by editing the configuration parameters in *Integration Server_directory \instances\instance_name\config\server.cnf*.

Before you disable or re-enable locking, complete the following:

- Make sure that all users have completed development on the server and unlocked all elements.
- Close all Designer sessions. After you change the extended settings in the following procedure, users will need to open a new Designer session.

Disabling Locking

To disable locking, you edit the extending settings using the Integration Server Administrator or manually edit *server.cnf*. The following procedure describes the Integration Server Administrator procedure.

Important:

Make sure that you only use this method of changing the settings. Later, if you change the settings by editing *server.cnf*, conflicts can occur.

➤ **To disable locking on the Integration Server**

1. Complete the tasks in [“Disabling and Re-enabling Locking”](#) on page 822.
2. In the Integration Server Administrator, under **Settings**, click **Extended**.
3. Click **Edit Extended Settings**.
4. In the Extended Settings box, type a key and value according to the following table.

If you want to	Type this
Disable user locking and show no locks	watt.server.ns.lockingMode=none
Disable user locking but show system locks	watt.server.ns.lockingMode=system

5. Click **Save Changes**. The information is saved to *Integration Server_directory* \instances\ *instance_name* \config\server.cnf.
6. Restart the Integration Server. The updated settings are now in effect.

Re-Enabling Locking

To re-enable locking, you use the Integration Server Administrator or manually edit server.cnf. The following procedure describes the Integration Server Administrator procedure.

Important:

Make sure that you only use this method of changing the settings. Later, if you change the settings by editing server.cnf, conflicts can occur.

➤ To re-enable locking on the Integration Server

1. Complete the tasks in [“Disabling and Re-enabling Locking”](#) on page 822.
2. In the Integration Server Administrator, under **Settings**, click **Extended**.
3. Click **Edit Extended Settings**.
4. In the Extended Settings box, set the value of watt.server.ns.lockingMode to **full**.
5. Click **Save Changes**. The information is saved to *Integration Server_directory* \instances\ *instance_name* \config\server.cnf.
6. Restart the Integration Server for the changes to take effect.

Best Practices

Remote Server Configuration

- It is not recommended that you use Cooperative Development functionality in an Integration Server cluster. Locking information for elements could be inadvertently shared with another Integration Server in the cluster. Use a standalone Integration Server not a cluster, while developing to eliminate these Cooperative Development problems.

Server User Names

- When logging on to the Integration Server, use a distinct user name. Locking is based on your user name, so it is important that each user log on to the server with a unique user name (not “Administrator” or “Developer”).

Package Replication and Publishing

- Always back up your packages every day or night using package replication and publishing. Because locking information does not travel with packages (or partial packages) when they are replicated, it is recommended that you apply a version to each package according to date. Do not replace or overwrite packages; delete the old package entirely and then install the new package.

Note:

If you do replace or overwrite packages, webMethods Integration Server takes the intersection of elements in the Navigation panel. It will also move the existing package to the `replicate\salvage` folder of the server instance.

- When you replicate and publish a package, the locking information is not preserved. This is expected behavior and is part of the feature’s design. You can, however, preserve system locks (read-only file attributes).
- Before you publish a package, keep in mind that user locks are not preserved.
- When you salvage a deleted package, lock information is not preserved. Before you salvage or delete a package, make sure that all locks are removed from the destination package.

It is not recommended that you use system or user locking on packages that are frequently replicated and/or partially replicated. For example, when sending frequently updated packages to partners.

Package and Folder Organization

- Use a single package or folder per developer or per Java/C service.

Source Code

- If there has been a significant change to the source code, always reload the package to reflect the latest system locks.

Upgrading webMethods Integration Server

- When you upgrade the webMethods Integration Server to a new version, you lose all lock information. Therefore, before upgrading, make sure that all locks are removed and all changes are saved.

44 Managing webMethods Messaging Triggers

■ Introduction	828
■ Managing Document Retrieval	828
■ Managing Document Processing	838
■ Limiting Server Threads for webMethods Messaging Triggers	846
■ Cluster Synchronization for webMethods Messaging Trigger Management	848
■ Modifying webMethods Messaging Trigger Properties	851
■ Managing Trigger Service Retries and Shutdown Requests	852
■ Delaying Polling Requests for webMethods Messaging Triggers	853
■ Serial Triggers Migrated to Integration Server 10.3 or Later from Earlier Versions	857

Introduction

In a publish-and-subscribe solution, the retrieval and flow of documents through Integration Server consumes resources, primarily server threads and memory. The rate at which Integration Server can retrieve and process documents is determined by the availability of these resources. However, server resources also need to be available to perform other functions. Integration Server Administrator provides controls for managing resources consumed by document retrieval and document processing. You can use these controls to balance the resource demands for document retrieval and processing with the server resources needed to perform other work.

Specifically, you can use the controls provided by Integration Server Administrator to:

- Increase or decrease the number of server threads used to retrieve documents from the Broker.
- Decrease the capacity of webMethods messaging trigger queues.
- Suspend document retrieval for one or more webMethods messaging triggers.
- Increase or decrease the number of server threads used to process documents.
- Decrease the number of threads that Integration Server can use to process documents for concurrent webMethods messaging triggers simultaneously.
- Suspend document processing for one or more webMethods messaging triggers.
- Change the configured trigger queue capacity, refill level, or execution threads for a specific webMethods messaging trigger.

Additionally, the Integration Server Administrator provides a cluster synchronization feature that you can use to propagate selected changes to other Integration Servers in a cluster automatically.

These controls can be useful in a production environment to free up server threads and memory to accommodate an unexpected server load (such as a sudden influx of HTTP requests) or in anticipation of a high usage time. You can also use the controls during the capacity planning stage of your project to determine the configured values for webMethods messaging triggers and server thread usage.

The following sections contain more information about managing document retrieval and document processing using the provided controls.

Note:

Prior to Integration Server version 9.5 SP1, webMethods messaging triggers were called Broker/local triggers.

Managing Document Retrieval

Within the publish-and-subscribe model, document retrieval is the process in which Integration Server uses a server thread to fetch more documents from the messaging provider. Document retrieval requires a server thread with which to request and retrieve documents from the messaging provider. Document retrieval also requires memory because Integration Server keeps temporary

copies of the documents it is retrieving in memory. Integration Server releases the temporary copies from memory after successfully processing the document.

Integration Server provides controls that you can use to manage the server resources used for document retrieval. Specifically, you can use the controls to:

- Manage the rate of document retrieval and the amount of memory used during document retrieval by adjusting trigger queue capacity.
- Suspend or resume document retrieval for one or more webMethods messaging triggers.
- Limit the number of server threads used for retrieving documents from Broker.

These controls can be used during development, capacity planning, or at run time. The following sections provide more information about these controls.

You can also configure the frequency with which Integration Server polls each trigger client queue on the Broker for new documents. For more information, see [“Delaying Polling Requests for webMethods Messaging Triggers” on page 853](#).

Increasing or Decreasing Threads for Document Retrieval from webMethods Broker

During production and capacity planning, you can increase or decrease the number of threads used to retrieve documents from the Broker. By default, Integration Server can use up to 100% of the server thread pool to retrieve documents from Broker. Each webMethods messaging trigger uses a separate server thread to retrieve documents from the Broker. For example, if the maximum size of the server thread pool is 80 threads, and the server can use 100% of the server thread pool to retrieve documents, then up to 80 triggers can request more documents at one time.

You can limit the maximum number of threads used for document retrieval by specifying the percentage of the server thread pool that can be used to retrieve documents. The Integration Server uses the specified percentage to calculate the number of server threads that can be used to retrieve documents from the Broker.

For example, suppose that the maximum size of the server thread pool is 80 threads. If you specify a maximum document retrieval threads percentage of 10%, then the Integration Server can use only 8 threads to retrieve documents from Broker at one time. Because Integration Server uses a separate thread to retrieve documents for each webMethods messaging trigger, this means that Integration Server can retrieve documents for only 8 triggers that receive documents from the Broker at one time.

Reducing the percentage of the server thread pool used for document retrieval from the Broker can slow the rate of document retrieval because fewer triggers can retrieve documents simultaneously. It also ensures the availability of server threads for other tasks, such as answering HTTP requests, retrieving documents from Universal Messaging, or processing documents.

Increasing the percentage of the server thread pool available for retrieving documents from Broker can increase the arrival rate of documents because it allows more triggers to retrieve documents from the Broker at one time.

For more information about setting the number of server threads for document retrieval, see [“Limiting Server Threads for webMethods Messaging Triggers” on page 846](#).

Note:

Threads allotted for document retrieval only impact webMethods messaging triggers that receiving documents from Broker. The threads you specify for document retrieval will not include threads used to retrieve documents from Universal Messaging

When to Increase or Decrease Threads for Retrieving Documents from webMethods Broker

Your knowledge of your integration solution is the best tool for determining when to increase and decrease thread usage for retrieving documents from Broker. For example, if you know that Integration Server regularly receives a high number of HTTP requests during a certain time period, you might want to decrease thread usage for retrieving documents right before the HTTP requests usually begin, then increase threads for retrieving documents after the frequency of HTTP requests slows down. Alternatively, if you know that Integration Server receives a high volume of incoming documents at the same time each day, you might want to increase the number of threads available for retrieving documents during that time period.

You can also determine when to increase or decrease threads for retrieving documents from Broker by monitoring the number of available server threads. To assist with this, you can establish a warning threshold that instructs Integration Server to alert you when the percentage of available threads drops below a specified level. Specifically, Integration Server creates a journal log entry stating "Available Thread Warning Threshold Exceeded." When you receive this message in the journal log, you can decrease threads for document retrieval to make server threads available to perform other functions. For more information about setting an available threads warning threshold, see [“Managing the Server Thread Pool” on page 109](#).

Another method of determining when to alter the number of server threads allotted for retrieving documents from Broker is to monitor the current number of threads retrieving documents from the Broker. Integration Server Administrator displays this value in the **Current Threads** field under **Document Retrieval** on the **Settings > Messaging > webMethods Messaging Trigger Management** page. The **Current Threads** field under **Document Retrieval** does not include threads used to retrieve documents from Universal Messaging.

Note:

Other ways to control the resources used for retrieving documents are: adjusting trigger queue capacity and suspending/resuming document retrieval for webMethods messaging triggers. For more information about adjusting trigger queue capacity, see [“Decreasing the Capacity of Trigger Queues” on page 830](#). For more information about suspending (or resuming) document processing, see [“Suspending and Resuming Document Retrieval” on page 833](#).

Decreasing the Capacity of Trigger Queues

You can impact the amount of memory used for document retrieval by adjusting the capacity of all the webMethods messaging triggers that retrieve documents from the Broker or Universal Messaging. The capacity determines the maximum number of documents that the trigger queues on Integration Server can contain.

You can use the **Queue Capacity Throttle** provided in Integration Server Administrator to decrease the capacity of all the trigger queues for webMethods messaging triggers that receive documents from a messaging provider. The throttle also reduces the refill levels of all webMethods messaging triggers that receive documents from the Broker. The refill level specifies the number of documents that remain in the trigger queue before Integration Server requests more documents. The **Queue Capacity Throttle** reduces the capacity and refill levels for all the trigger queues by the same percentage. For example, if you set the **Queue Capacity Throttle** to 50% of maximum, a trigger queue with a capacity of 10 and a refill level of 4 will have an adjusted capacity of 5 and an adjusted refill level of 2.

Note: webMethods messaging triggers that receive documents from Universal Messaging do not have a refill level.

By decreasing the capacity and refill levels, you can:

- Reduce the amount of memory needed to retrieve documents from the messaging provider.
Reduced capacity and refill levels mean that Integration Server retrieves fewer documents for a webMethods messaging trigger at one time. Because Integration Server retrieves fewer documents, Integration Server uses less memory when retrieving documents.
- Reduce the memory needed to store the documents while they await processing.

Note:

Decreasing the capacity might increase the frequency with which Integration Server retrieves documents because Integration Server might empty the trigger queues to the adjusted refill level more quickly.

Decreasing the Capacity and Refill Level for webMethods Messaging Triggers

Use the following procedure to use the **Queue Capacity Throttle** to decrease the capacity and refill levels for the queues of all webMethods messaging triggers.

Note:

The **Queue Capacity Throttle** does not affect the refill level of webMethods messaging triggers that receive documents from Universal Messaging because these triggers do not have a refill level.

➤ To decrease the capacity and refill level of trigger queues

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**, and then click **Edit Global webMethods Messaging Trigger Controls**.

4. Under **Document Retrieval**, in the **Queue Capacity Throttle** field, select the percentage of configured capacity at which you want all trigger queues to operate. The Integration Server automatically adjusts the refill levels by the same percentage.
5. If you want to apply the queue capacity throttle change to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs to a properly configured cluster and is configured to synchronize webMethods messaging trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

6. Click **Save Changes**.

Notes:

- The Queue Capacity Throttle only affects the refill level of webMethods messaging triggers that receive documents from Broker. webMethods messaging triggers that receive documents from Universal Messaging do not have a refill level.
- The **Queue Capacity Throttle** setting is maintained across server restarts and package reloads.
- If the percentage by which you reduce capacity does not resolve to a whole number, Integration Server rounds up or rounds down to the nearest whole number. However, if rounding down would reduce the value to 0, Integration Server rounds up to 1. For example, if you set the **Queue Capacity Throttle** to 10% of maximum, a trigger queue with a capacity of 15 and refill level of 4 will have an adjusted capacity of 2 and an adjusted refill level of 1 (Integration Server rounds the calculated adjusted capacity of 1.5 up to 2 and rounds the calculated adjusted refill level of 0.4 up to 1).
- When you reduce the **Queue Capacity Throttle** and save your changes, Integration Server does not immediately reduce the number of documents in a trigger queue. Instead, Integration Server continues to process documents in the trigger queue until it reaches the adjusted refill level. Then, Integration Server retrieves enough documents to fill the trigger queue to the adjusted capacity. For example, if you set **Queue Capacity Throttle** to 50%, a trigger queue with a capacity of 8 and a refill level of 2 will have an adjusted capacity of 4 and an adjusted refill level of 1. The Integration Server processes documents in the trigger queue until it reaches the adjusted refill level of only 1 document. Then, Integration Server retrieves up to 3 documents to increase the number of documents in the queue to 4 (the adjusted capacity).
- If you reduce the capacity to a low percentage for an extended period of time, the document might expire on the messaging provider. For each publishable document type, you can specify a **Time to live** property. This property specifies how long a document can remain on the messaging provider before the messaging provider discards it. For more information about publishable document types, see *webMethods Service Development Help*.
- If you use the **Queue Capacity Throttle** as part of your capacity planning process and you determine that the configured values for trigger capacity and refill level need to change, you can use the Integration Server Administrator or Software AG Designer to set the new capacity

and refill level values for each webMethods messaging trigger. For more information about setting the capacity and refill level for a trigger, see [“Modifying webMethods Messaging Trigger Properties” on page 851](#).

Suspending and Resuming Document Retrieval

You can reduce the amount of server resources that document retrieval consumes by suspending document retrieval for one or more webMethods messaging triggers. Using the Integration Server Administrator, you can:

- Suspend or resume document retrieval for all webMethods messaging triggers.
- Suspend or resume document retrieval for specific webMethods messaging triggers.

The following sections provide more information about these options.

About Suspending and Resuming Document Retrieval for All Triggers

When you suspend document retrieval for all the webMethods messaging triggers on an Integration Server, the Integration Server stops retrieving documents from the Broker and Universal Messaging. Integration Server resources, such as threads and memory, that would have been used for document retrieval are available for other tasks.

Suspending document retrieval for all webMethods messaging triggers is a quick way to free up server resources. This can be especially helpful in a situation in which the Integration Server is functioning under heavy load and additional resources are needed immediately.

Suspending or resuming document retrieval can be a temporary or permanent change. (Integration Server considers a document retrieval change to be permanent if you selected the **Apply Change Permanently** check box when you made the change.) If the change is temporary, Integration Server reverts to the permanent document retrieval state when Integration Server restarts or you reload a package. When you reload a package, Integration Server reverts only the webMethods messaging triggers contained in that package to the permanent document retrieval state. For example, suppose that you temporarily suspend document retrieval for all webMethods messaging triggers. If you reload the package OrderProcessing, Integration Server resumes document retrieval for the webMethods messaging triggers in the OrderProcessing package only.

Tip:

On the **Settings > Messaging > webMethods Messaging Trigger Management** page, under **Document Retrieval**, Integration Server Administrator indicates a temporary document retrieval change by displaying an asterisk (*) next to the trigger status in the **Active** column.

You can gradually resume document retrieval by setting the **Queue Capacity Throttle** to a low percentage, such as 10%, and then resuming document retrieval for all webMethods messaging triggers. Integration Server resumes document retrieval at the adjusted capacity for all webMethods messaging triggers. You can also gradually resume document retrieval by selectively resuming individual triggers. For example, you might want to first resume document retrieval for those triggers that are part of a critical or high priority process.

Before suspending or resuming document retrieval, keep the following information in mind:

- Suspending document retrieval affects document retrieval from the messaging provider only (Broker and Universal Messaging). webMethods messaging trigger will continue to receive locally published documents. Additionally, webMethods messaging triggers will continue to receive documents delivered to the default client.
- When you suspend document retrieval, documents to which this trigger subscribes will collect on the messaging provider. Documents remain on the messaging provider until document retrieval resumes for the trigger or the documents expire.
- If you suspend document retrieval for a trigger, but do not suspend document processing for the trigger, one of the following occurs:
 - If Broker is the messaging provider, the trigger eventually processes all the documents that were retrieved from the messaging provider for the trigger
 - If Universal Messaging is the messaging provider, Integration Server rolls back all of the documents that have been retrieved but not acknowledged to Universal Messaging. This includes unprocessed documents in the trigger queue and documents currently being processed. The documents that are in the midst of being processed will complete processing. However, Integration Server cannot acknowledge the documents. This will lead to duplicates. After document processing resumes for the trigger, the documents are redelivered. If exactly-once processing is enabled for the trigger but a message history database is not used, Integration Server determines that the redelivered documents are duplicate documents and rejects them.

To avoid the above situation where Integration Server receives duplicates of documents for which processing completed, suspend document processing before suspending document retrieval. When you suspend document processing for a webMethods messaging trigger that receives documents from Universal Messaging, Integration Server does not retrieve any new documents from the trigger queue for processing. Integration Server allows any documents already being processed to complete processing, which includes acknowledging the documents to Universal Messaging. After processing completes, then suspend document retrieval for the trigger. An active thread count of zero for the trigger indicates that document processing has completed.

Note:

If Universal Messaging setting `QueueDeliveryPersistencePolicy` is set to “No Persist/No Sync” and the Universal Messaging server restarts before document retrieval resumes for the trigger, the documents that Integration Server rolled back to Universal Messaging will be lost.

- When you resume document retrieval, Integration Server resumes document retrieval for webMethods messaging triggers at the percentage specified by the **Queue Capacity Throttle**.
- If you do not resume document retrieval before Integration Server restarts, the trigger package reloads, or the trigger properties are modified, the messaging provider discards any volatile documents that it is holding for the trigger.

Suspending or Resuming Document Retrieval for All webMethods Messaging Triggers

Use the following procedure to suspend or resume document retrieval for webMethods messaging triggers.

➤ To suspend or resume document retrieval for all webMethods messaging triggers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**.
4. Under **Individual webMethods Messaging Trigger Controls**, in the **Active** column located under **Document Retrieval**, click **edit all**.
5. In the **Retrieval State** list, do the following:

Select	To
Active	Resume document retrieval for all webMethods messaging triggers.
Suspended	Suspend document retrieval for all webMethods messaging triggers.

6. If you want the state change to be permanent and maintained after Integration Server restarts or after a package reload, select the **Apply Change Permanently** check box. If you do not select this check box, Integration Server considers the change to be temporary.
7. If you want to apply the document retrieval change to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs is configured to synchronize trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

8. Click **Save Changes**.

Notes:

- Integration Server does not suspend (or resume) document retrieval if the webMethods messaging trigger is locked or disabled.
- If Integration Server cannot suspend (or resume) document retrieval on the local Integration Server, cluster synchronization cannot occur.

- Integration Server does not suspend (or resume) document retrieval for webMethods messaging triggers that have been excluded from trigger management changes using the `watt.server.trigger.managementUI.excludeList`. For more information about this property, see [“Server Configuration Parameters” on page 1017](#).

About Suspending and Resuming Document Retrieval for a Specific Trigger

Sometimes, instead of suspending or resuming document retrieval for all webMethods messaging triggers, you might want to suspend or resume document retrieval for specific webMethods messaging triggers. Following are some situations in which you might want to suspend or resume document retrieval for specific triggers.

- When a back-end system needs maintenance or is becoming unresponsive, you might want to suspend document retrieval for triggers that interact with the back-end system. By suspending document retrieval, documents that would normally accumulate on Integration Server awaiting processing remain on the messaging provider. This keeps memory and other server resources available for other activities. When the back-end system becomes available, you can resume document retrieval for the associated triggers.
- After suspending document retrieval for all triggers, you might resume document retrieval for specific triggers. If Integration Server is functioning under an unusually heavy load, you might first suspend retrieval for all triggers and then gradually resume retrieval, starting with the triggers involved in key processes.
- If Integration Server suspends document retrieval for a serial trigger because the associated trigger service ends in error, you need to resume document retrieval for that trigger. For more information about configuring a serial trigger to suspend retrieval and processing automatically after an error, see *webMethods Service Development Help*.

Suspending or Resuming Document Retrieval for a webMethods Messaging Trigger

The following procedure explains how to suspend or resume document retrieval for an individual webMethods messaging trigger.

➤ To suspend or resume document retrieval for a webMethods messaging trigger

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**.
4. Under **Individual webMethods Messaging Trigger Controls**, locate the row containing the trigger for which you want to suspend document retrieval.
5. In the **Active** column located under **Document Retrieval**, click **Yes** or **No**. (The **Active** column displays "Yes" if document retrieval is active for the trigger; "No" if document retrieval is

suspended. An asterisk (*) appears next to the status if the document retrieval state is temporary.)

- In the **Retrieval State** list, do the following:

Select	To
Active	Resume document retrieval for this trigger.
Suspended	Suspend document retrieval for this trigger.

- If you want the state change to be permanent and maintained after Integration Server restarts, select the **Apply Change Permanently** check box. If you do not select this check box, Integration Server considers the change to be temporary.
- If you want to apply the document retrieval change for this trigger to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs to a properly configured cluster and is configured to synchronize trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

- Click **Save Changes**.

Notes:

- Integration Server will not suspend or resume document retrieval for the specified webMethods messaging trigger if the trigger is locked by a user.
- If Integration Server cannot suspend (or resume) document retrieval locally, cluster synchronization cannot occur.
- When you resume document retrieval, Integration Server resumes retrieval for the webMethods messaging trigger at the percentage specified by the **Queue Capacity Throttle**.
- In a flow service, you can suspend or resume document retrieval for individual triggers by invoking the `pub.trigger:suspendRetrieval` service or the `pub.trigger:resumeRetrievalService`, respectively. For more information about these services, see the *webMethods Integration Server Built-In Services Reference*.
- In a Java service, you can suspend or resume document retrieval by calling `com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade.setRetrievalSuspended()`. For more information about this method, see the *webMethods Integration Server Java API Reference* for the `com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade` class.

- You can filter the list of displayed triggers using the `watt.server.trigger.managementUI.excludeList` property. For more information about this property, see [“Server Configuration Parameters” on page 1017](#).

Managing Document Processing

Within the publish-and-subscribe model, *document processing* is the process of evaluating documents against trigger conditions and executing the appropriate trigger services to act on those documents. Document processing requires a server thread with which to evaluate the document and execute the trigger service. It also requires memory in which to keep a copy of the document during document evaluation and trigger service execution.

Integration Server provides various controls that you can use to manage the rate of and resource demands for document processing. Specifically, you can:

- Limit the number of server threads used by webMethods messaging triggers for processing documents.
- Manage the number of server threads that can be used to process documents for a webMethods messaging trigger concurrently.
- Suspend or resume document processing for one or more webMethods messaging triggers.

These controls can be used as part of your capacity planning or used during production. The following sections provide more information about these controls.

Increasing or Decreasing Threads for Processing Documents Received by webMethods Messaging Triggers

During production and capacity planning, you can increase or decrease the number of threads that can be used to simultaneously process documents received by webMethods messaging triggers. The number of threads available for processing documents helps determine the rate at which Integration Server processes documents. By default, Integration Server can use up to 100% of the server thread pool to process documents (execute triggers). Each time Integration Server processes a document, Integration Server uses a server thread. For example, if the maximum size of the server thread pool is 80 threads, and Integration Server can use 100% of the server thread pool to execute triggers, then up to 80 triggers can execute at the same time. That is, Integration Server can process up to 80 documents simultaneously.

You can control the number of server threads available for document processing (trigger execution) by specifying the maximum percentage of the server thread pool that can be used to process documents. Integration Server uses the percentage to calculate the number of server threads that can be used for trigger execution (document processing). For example, suppose that the maximum size of the server thread pool is 80 threads. If you set 10% as the maximum percentage of document processing threads, then Integration Server can use up to 8 threads simultaneously to execute webMethods messaging triggers.

By reducing the number of server threads used for processing documents, you can make server threads and memory available to perform other tasks, such as executing HTTP requests and retrieving documents. Alternatively, you can increase the number of threads for document

processing to allow the server to process more documents simultaneously. This can also allow the server to drain the trigger queues more quickly and request additional documents more frequently.

Document processing for serial and concurrent triggers combined cannot exceed the value determined by the maximum document processing threads percentage. If you reduce the percentage of document processing server threads, and concurrent triggers continue to consume the maximum execution threads possible (according to their configured settings), serial triggers must wait longer for server threads to become available. This is especially likely if Integration Server contains concurrent triggers that execute long-running services.

For more information about setting the number of server threads for document processing, see [“Limiting Server Threads for webMethods Messaging Triggers” on page 846](#).

Tip:

If you decrease the percentage of threads that can be used for document processing, consider decreasing the **Execution Threads Throttle** to prevent concurrent triggers from monopolizing available server threads.

When to Increase or Decrease Threads for Processing Documents

Your knowledge of your integration solution is the best tool for determining when to adjust thread usage for document processing (trigger execution). For example, suppose that a batch process that occurs at the same time each day results in a spike in document publishing. You might want to increase threads for document processing right before the batch process starts to make server threads available to process documents.

Alternatively, if you observe memory constraints or other resource issues, you can decrease the number of threads for document processing. Document processing consumes memory because Integration Server keeps the document in memory while the server thread evaluates the document and executes the trigger service.

Note:

You can also determine when to modify the number of threads allowed for document processing by monitoring thread usage. You can do this by viewing the thread usage information displayed on the **Server > Statistics** page. However, you can also establish a warning threshold that instructs Integration Server to alert you when the number of available threads drops below a particular level. Specifically, Integration Server creates a journal log entry stating "Available Threads Warning Threshold Usage Exceeded." When Integration Server writes this journal log entry, you might want to decrease threads for document processing to allow more threads to be used for other functions. For more information about setting an available threads warning threshold, see [“Managing the Server Thread Pool” on page 109](#).

Another way to determine when to alter the number of server threads allotted for processing documents is to monitor the current number of threads that are processing documents for triggers. Integration Server Administrator displays this value in the **Current Threads** field located under **Document Processing** on the **Settings > Messaging > webMethods Messaging Trigger Management** page.

Note:

Other ways to control the resources used for document processing are: adjusting execution threads for concurrent triggers and suspending or resuming document processing for triggers. For more information about adjusting trigger queue capacity, see [“Decreasing Document Processing for Concurrent Triggers” on page 840](#). For more information about suspending (or resuming) document processing, see [“Suspending and Resuming Document Processing” on page 842](#).

Decreasing Document Processing for Concurrent Triggers

You can reduce the amount of server resources consumed by document processing by decreasing the rate of processing for concurrent triggers. Specifically, you can reduce the maximum number of threads that can process documents for a concurrent trigger at one time.

Integration Server Administrator provides an **Execution Threads Throttle** that you can use to reduce the execution threads for all concurrent triggers by the same percentage. For example, if you set the **Execution Threads Throttle** to 50% of maximum, Integration Server reduces the maximum execution threads for all concurrent triggers by half. A concurrent trigger with a maximum execution threads value of 6, has an adjusted maximum execution threads value of 3.

By decreasing parallel processing for concurrent triggers, you can:

- Free up server threads and memory to perform other functions, such as answering HTTP requests or retrieving documents.
- Prevent concurrent triggers from monopolizing the threads allotted for document processing. The number of server threads that Integration Server dispatches to process documents for serial and concurrent triggers cannot exceed the value established by the maximum execution threads percentage. If you reduce the number of threads allowed for document processing, and concurrent triggers continue to consume the maximum execution threads possible (according to their configured settings), serial triggers must wait longer for server threads to become available. This is especially likely if Integration Server contains concurrent triggers that execute long-running services.

Note:

The **Execution Threads Throttle** only affects webMethods messaging triggers.

Decreasing Execution Threads for Concurrent webMethods Messaging Triggers

➤ To decrease execution threads for concurrent webMethods messaging triggers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**, and then click **Edit Global webMethods Messaging Trigger Controls**.

4. Under **Document Processing**, in the **Execution Threads Throttle** field, select the percentage of the configured maximum execution threads value at which you want the server to function. Integration Server applies this percentage to the maximum execution threads value for all concurrent webMethods messaging triggers.
5. If you want to apply the **Execution Threads Throttle** change to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs to a properly configured cluster and is configured to synchronize trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

6. Click **Save Changes**.

Notes:

- The **Execution Threads Throttle** value is maintained across server restarts and package reloads.
- Serial triggers always process one document at a time. The **Execution Threads Throttle** property does not affect serial triggers.
- If the percentage by which you reduce trigger execution threads does not resolve to a whole number, Integration Server rounds up or rounds down to the nearest whole number. However, if rounding down would set the value to 0, Integration Server rounds up to 1. For example, if you reduce **Execution Threads Throttle** to 10% of maximum, a concurrent trigger with a maximum execution threads value of 12 would have an adjusted value of 1 (Integration Server rounds 1.2 down to 1). A concurrent trigger with a maximum execution threads value of 4 would have an adjusted value of 1 (Integration Server rounds 0.4 up to 1).
- When you reduce the **Execution Threads Throttle** and save your changes, Integration Server does not terminate threads currently executing concurrent triggers to meet the adjusted maximum. Integration Server allows server threads processing documents for concurrent triggers to execute to completion. Integration Server waits until the number of threads executing for a concurrent trigger is less than the adjusted maximum before dispatching another server thread to process a document for that trigger.
- If you suspend document processing (trigger execution) and do not suspend document retrieval, Integration Server will fill all the trigger queues to capacity. Full trigger queues consume more memory than empty trigger queues.
- You can also reduce the number of concurrent execution threads for a trigger by reducing the capacity of a trigger queue below the maximum number of concurrent execution threads for the trigger. The maximum number of dispatched threads for a trigger cannot exceed the trigger queue's capacity. For more information about reducing trigger queue capacity, see [“Decreasing the Capacity of Trigger Queues” on page 830](#).

- If you use the **Execution Threads Throttle** as part of your capacity planning process and you determine that the configured values for **Maximum execution threads** need to change, you can use the Integration Server Administrator or Software AG Designer to set the new maximum execution threads values for each concurrent trigger. For more information about setting trigger properties, see [“Modifying webMethods Messaging Trigger Properties” on page 851](#).

Suspending and Resuming Document Processing

You can reduce the amount of server resources that document processing consumes by suspending document processing for one or more webMethods messaging triggers. Using the Integration Server Administrator, you can:

- Suspend or resume document processing for all webMethods messaging triggers.
- Suspend or resume document processing for specific webMethods messaging triggers.

The following sections provide more information about these options.

About Suspending and Resuming Document Processing for all Triggers

When you suspend document processing for all webMethods messaging triggers, Integration Server stops dispatching server threads to process documents received by webMethods messaging triggers. Server resources, such as threads and memory, that might have been used for document processing will be available for other tasks. Document processing remains suspended until you specifically resume it.

Suspending document processing for all webMethods messaging triggers is a quick way to make server resources available. This can be especially helpful in a situation in which Integration Server is functioning under heavy load and additional resources need to be available immediately.

Suspending or resuming document processing can be a temporary or permanent change. (Integration Server considers a document processing change to be permanent if you selected the **Apply Change Permanently** check box when you made the change.) If the change is temporary, Integration Server reverts to the permanent document processing state when Integration Server restarts or you reload a package. When you reload a package, Integration Server reverts only the triggers contained in that package to the permanent document processing state. For example, suppose that you temporarily suspend document processing for all triggers. If you reload the package OrderProcessing, Integration Server resumes document processing for the triggers in the OrderProcessing package only.

Tip:

On the **Settings > Messaging > webMethods Messaging Trigger Management** page, under **Document Processing**, Integration Server Administrator indicates a temporary document processing change by displaying an asterisk (*) next to the trigger status in the **Active** column.

When you are ready to resume document processing, you might want to resume it gradually. For example, you might set the **Execution Threads Throttle** to a low percentage, resume document processing for all triggers, and then gradually move the **Execution Threads Throttle** up to 100%. Alternatively, you might selectively resume individual triggers. For example, you might want to resume document processing for those triggers that are part of a critical or high priority process.

Suspending or Resuming Document Processing for All webMethods Messaging Triggers

Use the following procedure to suspend or resume document processing for all webMethods messaging triggers.

➤ To suspend or resume document processing for all webMethods messaging triggers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**.
4. Under **Individual webMethods Messaging Trigger Controls**, in the **Active** column located under **Document Processing**, click **edit all**.
5. In the **Processing State** list, do the following:

Select	To
Active	Resume document processing for all of the webMethods messaging triggers.
Suspended	Suspend document processing for all of the webMethods messaging triggers.

6. If you want the state change to be permanent and maintained after Integration Server restarts, select the **Apply Change Permanently** check box. If you do not select this check box, the Integration Server considers the change to be temporary.
7. If you want to apply the document processing change to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs to a properly configured cluster and is configured to synchronize trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

8. Click **Save Changes**.

Notes:

- Integration Server will not suspend or resume document processing for a locked or disabled trigger.

- If Integration Server cannot suspend (or resume) document processing locally, cluster synchronization cannot occur.
- Integration Server does not suspend (or resume) document processing for webMethods messaging triggers that have been excluded from trigger management changes using the `watt.server.trigger.managementUI.excludeList`. For more information about this property, see [“Server Configuration Parameters” on page 1017](#)
- Suspending or resuming document processing affects all documents in the trigger's queue on Integration Server, including documents retrieved from the messaging provider and from local publishing.
- When you suspend document processing, Integration Server will not dispatch any more server threads to process documents. Any server threads currently processing documents for triggers will execute to completion. This includes documents that are being retried.
- When you suspend document processing but do not suspend document retrieval, documents will collect in trigger queues until the trigger queues are at maximum capacity or document processing resumes. If Integration Server restarts before document processing resumes, volatile documents are discarded.
- When you resume document processing, Integration Server resumes document processing for webMethods messaging trigger that at the percentage specified by the **Execution Threads Throttle**.

About Suspending and Resuming Document Processing for Specific Triggers

Sometimes, instead of suspending or resuming document processing for all webMethods messaging triggers, you might want to suspend or resume processing for a specific trigger. Following are examples of situations where you might want to suspend document processing for specific triggers.

- When a back-end system becomes unresponsive or requires maintenance, you might want to suspend document processing for triggers that interact with that back-end system. If the back-end system is not available because of maintenance or failure, trigger services that interact with the system would probably not execute successfully. Suspending document processing for the associated triggers allows for more effective resource utilization because you keep resources that would have been used for unsuccessful document processing available for other tasks.
- After suspending document processing for all triggers, you might resume document processing for specific triggers. If Integration Server is operating under a heavy load, you might first suspend all document processing and then gradually resume document processing, starting with the triggers involved in critical processes.
- If Integration Server suspends document processing for a serial trigger because the associated trigger service ends in error, you need to resume document processing for the trigger. For more information about configuring a serial trigger to suspend retrieval and processing automatically after an error, see *webMethods Service Development Help*.

Suspending or Resuming Document Processing for a Specific webMethods Messaging Trigger

Use the following procedure to suspend or resume document processing for an individual webMethods messaging trigger.

➤ To suspend or resume document processing for a webMethods messaging trigger

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**.
4. Under **Individual webMethods Messaging Trigger Controls**, locate the row containing the trigger for which you want to suspend document processing.
5. In the **Active** column located under **Document Processing**, click **Yes** or **No**. (The **Active** column displays "Yes" if document processing is active for the trigger; "No" if document processing is suspended. An asterisk (*) appears next to the status if the document processing state is temporary.)
6. In the **Processing State** list, do the following:

Select	To
Active	Resume document processing for this webMethods messaging trigger.
Suspended	Suspend document processing for this webMethods messaging trigger.

7. If you want the state change to be permanent and maintained after the Integration Server restarts, select the **Apply Change Permanently** check box. If you do not select this check box, the Integration Server considers the change to be temporary.
8. If you want to apply the document processing change for this trigger to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs to a properly configured cluster and is configured to synchronize trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

9. Click **Save Changes**.

Notes:

- Integration Server will not suspend or resume document processing for the specified webMethods messaging trigger if the trigger is locked by a user.
- If Integration Server cannot suspend (or resume) document processing locally, cluster synchronization cannot occur.
- When you resume document processing, the **Execution Threads Throttle** determines the maximum number of documents that can be processed in parallel.
- In a flow service, you can suspend or resume document processing for an individual webMethods messaging trigger by invoking the `pub.trigger:suspendProcessing` service or the `pub.trigger:resumeProcessing` service, respectively. For more information about these services, see the *webMethods Integration Server Built-In Services Reference*.
- In a Java service, you can suspend or resume document retrieval by calling `com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade.setProcessingSuspended()`. For more information about this method, see the *webMethods Integration Server Java API Reference* for the `com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade` class.
- You can filter the list of displayed triggers using the `watt.server.trigger.managementUI.excludeList` property. For more information about this property, see [“Server Configuration Parameters” on page 1017](#).

Limiting Server Threads for webMethods Messaging Triggers

Integration Server provides parameters that you can use to limit how many threads in the server thread pool retrieve and process documents for webMethods messaging triggers. You can specify what percentage of the server thread pool can be used to retrieve documents from the messaging provider. You can also specify the percentage of the server thread pool that can be used to process documents received by triggers.

For example, suppose that the server thread pool can contain up to 80 threads. If you specify that the maximum threads for document retrieval is 50% of the server thread pool, then Integration Server can use up to 40 threads for retrieving documents from the messaging provider.

Limiting the number of server threads that can retrieve and process documents from the messaging provider keeps server threads available to perform other server functions such as responding to http requests or server administration. These parameters help to ensure that retrieving and processing documents from the messaging provider will not monopolize the entire server thread pool.

Note:

When Integration Server uses a thread for document retrieval, the thread retrieves documents for one trigger from the messaging provider. The thread does not retrieve documents for all triggers. A thread used for document processing addresses one document in a trigger queue. (Document processing includes determining which trigger condition the document satisfies and executing the associated service.)

Setting the Maximum Number of Server Threads for webMethods Messaging Triggers

Keep the following points in mind when specifying the number of threads for retrieving and processing documents for a webMethods messaging trigger:

- The Document Retrieval maximum threads percentage impacts webMethods messaging trigger that retrieve documents from Broker only.
- The Document Processing maximum threads percentage impacts all webMethods messaging triggers.

➤ To set the maximum number of server threads for webMethods messaging triggers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**, and then click **Edit Global webMethods Messaging Trigger Controls**.
4. Under **Document Retrieval**, in the **Maximum Threads** field, type the maximum percentage of the server thread pool that can be used to retrieve documents from the Broker. You must enter a value greater than zero. The default is 100%.

If the **Maximum Threads** field under Document Retrieval displays "(Broker Not Configured)", then this Integration Server is not configured to connect to a Broker.

5. Under **Document Processing**, in the **Maximum Threads** field, type the maximum percentage of the server thread pool that can be used to process documents received by webMethods messaging triggers. You must enter a value greater than zero. The default is 100%.
6. Click **Save Changes**.

Notes:

- Integration Server uses the percentages you entered to calculate the number of threads that can be devoted to retrieving and processing documents. If the number of threads does not evaluate to a whole number, Integration Server rounds up or down to the nearest whole number.
- For document retrieval, if the current number of server threads retrieving documents is greater than the new value set by the **Maximum Threads** percentage, Integration Server will not dispatch more threads for retrieving documents from the Broker. Threads currently retrieving documents will execute to completion. Integration Server will dispatch new threads for retrieving documents from the Broker only when the current number of document retrieval threads is less than the maximum allowed document retrieval threads.

- For document processing, if the current number of server threads processing documents (executing triggers) is greater than the thread value determined by the **Maximum Threads** percentage, Integration Server will not dispatch more threads for document processing. Threads currently processing documents will execute to completion. Integration Server will dispatch new threads for trigger execution only when the current number of document processing threads is less than the maximum allowed document processing threads.
- The current number of threads and maximum allotted threads for retrieving documents from the Broker is displayed in the **Current Threads** field under Document Retrieval on the **Settings > Messaging > webMethods Messaging Trigger Management** page.
- The current number of threads and maximum allotted threads for processing documents is displayed in the **Current Threads** field under Document Processing on the **Settings > Messaging > webMethods Messaging Trigger Management** page.

Cluster Synchronization for webMethods Messaging Trigger Management

If the Integration Server is a member of a cluster, changes that you make for document retrieval, document processing, and for trigger properties can be propagated to other servers in the cluster automatically. Propagating the changes to other servers in the cluster prevents you from needing to make identical changes on all the servers manually. It can also prevent the other servers from absorbing the resource demands that would have been directed to the first server.

Note:

Trigger management changes made using the pub.trigger services can also be applied across a cluster.

Configuring Cluster Synchronization

Integration Server propagates trigger management changes to other members of a cluster by performing a remote invoke. For cluster synchronization to succeed, you need to complete the following tasks:

- Configure the cluster. Integration Server can propagate trigger management changes to other servers only if all the servers are members of a properly configured cluster. For more information about configuring a cluster, see the *webMethods Integration Server Clustering Guide*.
- Set up remote server aliases for the servers in the cluster. For more information about setting up aliases for remote servers, see “Setting Up Aliases for Remote Integration Servers” on page 115.
- Update the `watt.server.cluster.aliasList` property with a comma-separated list of the remote server alias names. Integration Server uses this list when executing the remote invokes that update the other cluster nodes.

Note: Integration Servers that belong to the cluster but are not in this list will not be updated during cluster synchronization.

When cluster synchronization property `watt.server.cluster.aliasList` is properly configured, the **Apply Change Across Cluster** check box will be visible when performing trigger management tasks.

Cluster Synchronization at Run Time

At run time, Integration Server uses remote invokes to update the other members of a cluster with trigger management changes. If a remote invoke to a server fails or that server is not available at the time of the remote invoke, the cluster will be out of sync. Integration Server executing the remote invoke displays the following journal log messages to indicate the status of the cluster synchronization attempt.

The following table identifies the log message Integration Server writes for each synchronization attempt result.

If synchronization	Expect this log message
Succeeds	Integration Server Administrator displays the message: <pre data-bbox="553 831 1461 863">Settings changed successfully.</pre>
Succeeds for some servers, but fails for others	Integration Server Administrator displays the message: <pre data-bbox="553 951 1461 1035">[ISS.0085.9203] Errors occurred while updating remote aliases (x of y updates failed). See server logs for more details.</pre> <p data-bbox="553 1073 1477 1136">The server log displays the message for each member of the cluster that was not successfully updated:</p> <pre data-bbox="553 1161 1461 1245">[ISS.0098.0107E] Error occurred during cluster invoke: Alias = remoteAliasName; Service = serviceName; Exception = exceptionName</pre> <p data-bbox="553 1283 1477 1346">You can use the Integration Server Administrator to view and change cluster synchronization status for triggers.</p>
Fails because the local update failed	Integration Server Administrator displays the message: <pre data-bbox="553 1440 1477 1535">[ISS.0085.9204] Local update failed: Exception providing reason for failure. (Note: The cluster synchronization will not run until all local errors are resolved.)</pre> <p data-bbox="553 1566 1477 1797">If the trigger management change cannot be completed on the local Integration Server, cluster synchronization cannot occur. For example, if you suspend document retrieval for all triggers and one trigger is currently locked, the Integration Server can suspend document retrieval for every trigger except the locked one. Because document retrieval could not be completed locally, the Integration Server cannot synchronize the change with the rest of the cluster.</p>

If synchronization	Expect this log message
Fails because it is not configured	<p>The server log displays the following message:</p> <pre data-bbox="462 304 1364 399">[ISS.0033.0156W] Cluster invoke did not complete successfully. Cluster Synchronization feature is not configured.</pre> <p>For more information about configuring cluster synchronization for triggers, see “Configuring Cluster Synchronization” on page 848.</p>

Monitoring Cluster Synchronization

Integration Server Administrator provides a cluster view that you can use to see the trigger synchronization status across all the servers in the cluster. For each server listed in the `watt.server.cluster.aliasList` property, the cluster view indicates whether the other servers in the cluster (and its triggers) are in sync with current server. The cluster view is located on the **Settings > Messaging > webMethods Messaging Trigger Management > Cluster View** page.

Note:

The **Cluster View** page appears only if the current Integration Server belongs to a properly configured cluster and the server is configured to synchronize trigger changes across the cluster.

If a webMethods messaging trigger is not synchronized, the cluster view displays an error message that indicates how the trigger is out of sync with the trigger on the current server. For example, if document processing for a trigger is suspended locally, but active on another server in the cluster, the error message next to trigger name states "Processing State mismatch [local=suspended; remote=active]".

Integration Server considers a trigger on a remote server to be out of sync with the local trigger of the same name if either of the following is true:

- The triggers have different values for trigger queue capacity, refill level, or maximum execution threads.
- The triggers have different document retrieval or document processing states.

Note:

To log on to a remote server in the cluster, click the server alias in the **Remote Server Alias** column. When connecting, the remote server prompts you for user and password information. If you are connecting to the remote server via HTTPS and the HTTPS port requires certificates, you need to import a trusted certificate into the browser so that it can be presented at connection time. If the trusted certificates are not imported into the browser, when you try to connect to the remote server, you will receive a message informing you that the page is not available. For more information about client authentication and certificates, see [“Authenticating Clients” on page 515](#).

Modifying webMethods Messaging Trigger Properties

During capacity planning or production, you might decide that the configured trigger properties need to be reset. For example, you might determine that Integration Server performs document retrieval more smoothly for some triggers when the triggers' queue capacity operates at 80% of its configured value. Using Integration Server Administrator, you can reset the configured capacity for those triggers. In fact, any trigger property that affects thread or memory usage for document retrieval or document processing can be set using Integration Server Administrator. These properties include trigger queue capacity, refill level, and maximum execution threads.

The ability to change the capacity and refill level for a webMethods messaging trigger is limited to webMethods messaging trigger that receive documents from the Broker. The capacity for triggers that receive documents from Universal Messaging can be changed only at design time and not at run time. To set the capacity for a webMethods messaging trigger that receives documents from Universal Messaging, edit the trigger in Designer. Furthermore, webMethods messaging triggers that receive documents from Universal Messaging do not have a refill level.

➤ To modify webMethods messaging trigger properties

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **webMethods Messaging Trigger Management**.
4. Under **Individual webMethods Messaging Trigger Controls**, click the name of the trigger for which you want to edit properties.
5. Click **Edit Trigger Properties**.
6. Under **Properties**, do one or more of the following:

For this property	Specify
Queue Capacity	The maximum number of documents that the trigger queue can contain.
Queue Refill Level	The number of unprocessed documents that must remain in this trigger queue before Integration Server retrieves more documents for the queue from the Broker. The Queue Refill Level value must be less than or equal to the Queue Capacity value.

Note:

The ability to change the queue capacity at run time via Integration Server Administrator applies to webMethods messaging triggers that receive documents from Broker only. To set the capacity for a webMethods messaging trigger that receives documents from Universal Messaging, edit the trigger in Designer.

For this property	Specify
-------------------	---------

Note:

The **Queue Refill Level** applies to webMethods messaging triggers that receive documents from Broker only. The refill level does not apply to webMethods messaging trigger that receive documents from Universal Messaging. The **Queue Refill Level** field displays N/A for triggers that receive documents from Universal Messaging.

Max Execution Threads

The maximum number of documents that Integration Server can process concurrently. You can only specify a maximum execution threads value for concurrent triggers. This field displays N/A for serial triggers.

For more information and guidelines for setting trigger queue capacity, refill level, and maximum execution threads, see the *webMethods Service Development Help*

- If you want to apply the property changes for this trigger to all the servers in a cluster, select the **Apply Change Across Cluster** check box.

This check box appears only if the current Integration Server belongs to a properly configured cluster and is configured to synchronize trigger changes across the cluster. For more information about configuring an Integration Server to synchronize trigger management changes across a cluster, see [“Cluster Synchronization for webMethods Messaging Trigger Management” on page 848](#).

- Click **Save Changes**.

Note:

You can filter the list of displayed triggers using the `watt.server.trigger.managementUI.excludeList` property. For more information about this property, see [“Server Configuration Parameters” on page 1017](#).

Managing Trigger Service Retries and Shutdown Requests

While Integration Server retries a trigger service, Integration Server ignores requests to shut down the server until the trigger service executes successfully or all retry attempts are made. This allows Integration Server to process a document to completion before shutting down.

Sometimes, however, you might want Integration Server to shut down without completing all retries for trigger services. Integration Server provides the server parameter `watt.server.trigger.interruptRetryOnShutdown` that you can use to indicate that a request to shut down the Integration Server should interrupt the retry process for trigger services.

The following table describes the possible values and associated behavior for the `watt.server.trigger.interruptRetryOnShutdown` parameter.

Value	Description
false	<p>Indicates that Integration Server should not interrupt the trigger service retry process to respond to a shutdown request. Integration Server shuts down only after it makes all the retry attempts or the trigger service executes successfully. This is the default value.</p> <div data-bbox="553 426 1458 842" style="background-color: #f0f0f0; padding: 10px;"> <p>Important: If <code>watt.server.trigger.interruptRetryOnShutdown</code> is set to "false" and a trigger is set to retry until successful, a trigger service can enter into an infinite retry situation. If the transient error condition that causes the retry is not resolved, Integration Server continually re-executes the service at the specified retry interval. Because you cannot disable a trigger during trigger service execution and you cannot shut down the server during trigger service execution, an infinite retry situation can cause Integration Server to become unresponsive to a shutdown request. To escape an infinite retry situation, set the <code>watt.server.trigger.interruptRetryOnShutdown</code> to "true". The change takes effect immediately.</p> </div>
true	<p>Indicates that Integration Server should interrupt the trigger service retry process if a shutdown request occurs. Specifically, after the shutdown request occurs, Integration Server waits for the current service retry to complete. If the trigger service needs to be retried again (the service ends because of an <code>ISRuntimeException</code>), Integration Server stops the retry process and shuts down. Upon restart, the transport (the Broker, Universal Messaging, or, for a local publish, the transient store) redelivers the document to the trigger for processing.</p> <div data-bbox="553 1188 1458 1465" style="background-color: #f0f0f0; padding: 10px;"> <p>Note: If the trigger service retry process is interrupted and the transport redelivers the document to the trigger, the transport increases the redelivery count for the document. If the trigger is configured to detect duplicates but does not use a document history database or a document resolver service to perform duplicate detection, Integration Server considers the redelivered document to be "In Doubt" and will not process the document.</p> </div>

Note:

When you change the value of the `watt.server.trigger.interruptRetryOnShutdown` parameter, the change takes effect immediately.

Delaying Polling Requests for webMethods Messaging Triggers

You can conserve Integration Server and Broker resources used for retrieving documents by delaying polling requests when a webMethods messaging trigger is inactive. When a polling delay is in use, Integration Server waits the length of the polling delay before requesting more messages

from the Broker. You can configure the polling delay to gradually increase as the trigger continues to be inactive.

Note:

A webMethods messaging trigger is considered to be inactive if it did not receive any documents in its last request to the messaging provider.

During a polling delay, the webMethods messaging trigger will not retrieve any new documents. However, the webMethods messaging trigger will not use messaging provider resources during the delay nor will the webMethods messaging trigger be using Integration Server resources to poll the messaging provider.

To use a polling delay, you need to specify the following:

- The length of the delay that should elapse before the trigger polls for more messages for the webMethods messaging trigger. The `watt.server.control.triggerInputControl.delays` parameter determines the length of the delay. If you want to use a gradually increasing delay, you indicate the incremental delays by specifying a comma-separated list of integers. The `watt.server.control.triggerInputControl.delays` parameter is measured in milliseconds and has a default value of: 500, 1000, 1500, 5000
- The time interval at which Integration Server introduces the polling delay for an inactive webMethods messaging trigger. If the trigger remains inactive, the same time interval helps determine when Integration Server increases the delay between polling requests. The time interval is determined by the `watt.server.control.triggerInputControl.delayIncrementInterval` parameter. This parameter is measured in milliseconds and has a default value of 1000.

Integration Server uses the following formula to determine the polling delay for retrieving documents from the Broker for a webMethods messaging trigger:

$$\text{inactiveTime} / \text{interval} = \text{delay}$$

Where:

- *inactiveTime* is the number of milliseconds that the webMethods messaging trigger has been inactive. That is, the *inactiveTime* is the number of milliseconds that have elapsed since Integration Server last polled the Broker and the Broker had no messages for the trigger.
- *interval* is the value of the `watt.server.control.triggerInputControl.delayIncrementInterval`.
- *delay* is the result of the division rounded down to the nearest integer and corresponds to an index position in the comma-separated list of delay lengths specified for `watt.server.control.triggerInputControl.delays`.

Note:

If an exception occurs when parsing the supplied values for `watt.server.control.triggerInputControl.delays` or `watt.server.control.triggerInputControl.delayIncrementInterval`, Integration Server resets both configuration parameters to the default values.

How Integration Server Delays a Polling Request for a webMethods Messaging Trigger

To demonstrate how Integration Server polls the Broker for documents when a webMethods messaging trigger is inactive, a step-by-step example is provided in this section. This example relies on the default values for the related server configuration parameters, specifically:

- `watt.server.control.triggerInputControl.delayIncrementInterval = 10000`
- `watt.server.control.triggerInputControl.delays = 500,1000,1500,5000`

The following table describes each step that Integration Server takes as it polls the Broker for documents for an inactive webMethods messaging trigger.

Step	Description
1	<p>The webMethods messaging trigger becomes inactive.</p> <p>Integration Server considers a webMethods messaging trigger to be inactive if the trigger did not receive any documents in the last request to the Broker.</p>
2	<p>Integration Server begins to delay polling requests using the formula $inactiveTime/interval = delay$ to determine the length of the polling delay. Using the defaults for the server parameters as stated above,</p> <p>Integration Server determines the length of the polling delay using this equation:</p> $0/1000 = 0$ <p>The delay at the zero index position of the <code>watt.server.control.triggerInputControl.delays</code> property is 500. Integration Server will wait 500 milliseconds and then poll the Broker for more documents.</p>
3	<p>If the polling request does not return any documents for the webMethods messaging trigger, Integration Server determines the length of the polling delay using this equation:</p> $500/10000 = 0.05$ <p>Integration Server rounds the calculated delay of 0.05 down to 0. The delay at the zero index position of the <code>watt.server.control.triggerInputControl.delays</code> property is 500. Integration Server will wait 500 milliseconds and then poll the Broker for more documents.</p>
4	<p>If the trigger remains inactive Integration Server continues to determine the polling delay using the formula $inactiveTime/interval = delay$ where the <i>inactiveTime</i> increases by 500 milliseconds each time.</p>

- | Step | Description |
|------|---|
| 5 | <p>When the trigger has been inactive for 10000 milliseconds, Integration Server determines the length of the polling delay using this equation:</p> $10000/10000 = 1$ <p>The delay at the first index position of the <code>watt.server.control.triggerInputControl.delays</code> property is 1000. Integration Server will wait 1000 milliseconds and then poll the Broker for more documents.</p> |
| 6 | <p>If the trigger remains inactive Integration Server continues to determine the polling delay using the formula $inactiveTime/interval = delay$ where the <i>inactiveTime</i> increases by 1000 milliseconds each time.</p> <p>Note: Integration Server rounds the calculated <i>delay</i> down to the nearest integer. For example, when the trigger has been inactive for 15000 milliseconds, the length of the polling delay would be determined by this equation: $15000/10000 = 1.5$. Integration Server rounds this down to 1 and uses the polling delay in the second index position of the <code>watt.server.control.triggerInputControl.delays</code> property which is 1000 milliseconds.</p> |
| 7 | <p>When the trigger has been inactive for 20000 milliseconds, Integration Server determines the length of the polling delay using this equation:</p> $20000/10000 = 2$ <p>The delay at the second index position of the <code>watt.server.control.triggerInputControl.delays</code> property is 1500. Integration Server will wait 1500 milliseconds and then poll the Broker for more documents.</p> |
| 8 | <p>If the trigger remains inactive Integration Server continues to determine the polling delay using the formula $inactiveTime/interval = delay$ where the <i>inactiveTime</i> increases by 1500 milliseconds each time.</p> |
| 9 | <p>When the trigger has been inactive for 30000 milliseconds, Integration Server determines the length of the polling delay using this equation:</p> $30000/10000 = 3$ <p>The delay at the third index position of the <code>watt.server.control.triggerInputControl.delays</code> property is 5000. Integration Server will wait 5000 milliseconds and then poll the Broker for more documents.</p> |
| 10 | <p>If the trigger remains inactive Integration Server continues to determine the polling delay using the formula $inactiveTime/interval = delay$ where the <i>inactiveTime</i> increases by 5000 milliseconds each time.</p> |

Step	Description
	<p>Because 5000 is the largest value specified in the array for <code>watt.server.control.triggerInputControl.delays</code>, the polling delay will never be more than 5000 milliseconds. Integration Server uses this polling delay until the trigger becomes active.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The last value in <code>watt.server.control.triggerInputControl.delays</code> is the longest lag between the Broker placing the document in the client queue on the Broker and Integration Server retrieving the document.</p> </div>
11	<p>When a polling request to the Broker returns one or more documents, Integration Server considers the trigger to be active and stops using a polling delay.</p>

Serial Triggers Migrated to Integration Server 10.3 or Later from Earlier Versions

As of Integration Server version 10.3, when using Universal Messaging as the messaging provider, a webMethods messaging trigger with serial processing corresponds to a serial durable subscription on Universal Messaging. For Integration Server version 9.9 to 10.2, a webMethods messaging trigger with serial processing corresponded to a priority durable subscription on Universal Messaging. Prior to Integration Server 9.9, a webMethods messaging trigger with serial processing corresponded to a shared durable subscription on Universal Messaging.

All serial webMethods messaging triggers created on Integration Server 10.3 or later will correspond to a serial durable subscription. However, serial triggers migrated from Integration Server versions 9.9 to 10.2 will still correspond to a priority named object (durable subscription). Serial triggers migrated from an Integration Server prior to version 9.9 will still correspond to a shared named object (durable subscription). As a result, the serial trigger and the durable subscription will be out of sync. To synchronize the migrated serial trigger and the durable subscription, you must do one of the following:

- If you are using a fresh install of Universal Messaging 10.3 or later (that is, the Universal Messaging server was not migrated), when you start Integration Server, synchronize the publishable document types with the provider using Designer or the built-in service `pub.publish.syncToProvider`. Synchronizing the publishable document types causes Integration Server to reload the webMethods messaging triggers. Integration Server creates a serial durable subscription for each serial trigger.
- If you are using an installation of Universal Messaging 9.9 or later that was migrated from an earlier version, you must delete and recreate the durable subscription that corresponds to a serial trigger. For more information about deleting and recreating a durable subscription associated with a trigger, see [“Synchronizing the webMethods Messaging Trigger and Durable Subscription on Universal Messaging”](#) on page 858.

Synchronizing the webMethods Messaging Trigger and Durable Subscription on Universal Messaging

A webMethods messaging trigger and the associated durable subscription on Universal Messaging can get out of sync. For example, when you change the processing mode for a webMethods messaging trigger that uses a Universal Messaging connection alias that shares a client prefix, Integration Server *does not* delete and recreate the durable subscription that corresponds to the trigger on Universal Messaging. As a result, the trigger on Integration Server is out of sync with the durable subscription on Universal Messaging. To synchronize the webMethods messaging trigger and the associated durable subscription, you must delete and recreate the durable subscription.

Note:

A webMethods messaging trigger with a serial processing mode corresponds to a *serial* durable subscription on Universal Messaging. A webMethods messaging trigger with a concurrent processing mode corresponds to a *shared* durable subscription on Universal Messaging. For information about the type of durable subscription on Universal Messaging to which a serial trigger created on versions of Integration Server prior to 10.3 corresponds, see [“Serial Triggers Migrated to Integration Server 10.3 or Later from Earlier Versions” on page 857](#)

> To synchronize the webMethods messaging trigger and the durable subscription on Universal Messaging

- Do one of the following:
 - If the webMethods messaging trigger resides on the only Integration Server connected to Universal Messaging and the **Client Prefix Is Shared** property for the Universal Messaging connection alias is set to **No**, start the trigger to delete and recreate the corresponding durable subscription. You can start a trigger by disabling and then enabling the Universal Messaging connection alias used by the trigger.

Note: Integration Server starts triggers upon server restart.

- If more than one Integration Server connects to Universal Messaging or the **Client Prefix Is Shared** property for the Universal Messaging connection alias is set to **Yes**, you must use Universal Messaging Enterprise Manager to delete the durable subscription. Make sure to delete the durable subscription when the durable subscription is fully drained and no new documents will be sent to it. You may need to quiesce document publishers before deleting the durable subscription. Then create the durable subscription for the trigger by disabling and then enabling the Universal Messaging connection alias used by the trigger.

45 Managing JMS Triggers

■ Introduction to JMS Trigger Management	860
■ Searching for JMS Triggers	860
■ About JMS Trigger Status and State	861
■ Controlling Thread Usage for JMS Triggers	865
■ Configuring Integration Server Session Reuse	869
■ Configuring JMS Session Reuse	869
■ Delaying Polling Requests for Concurrent JMS Triggers	869
■ What Happens When JMS Triggers Fail to Start?	872
■ About WS Endpoint Triggers	875

45.1 Introduction to JMS Trigger Management

Integration Server and Integration Server Administrator provide ways for managing JMS triggers and the resources used by JMS triggers. Specifically, you can use the controls provided by Integration Server Administrator to:

- Increase or decrease the maximum number of server threads used for JMS triggers.
- Change the maximum execution threads for concurrent JMS triggers.
- Enable, disable, or suspend one or more JMS triggers.
- Delay the frequency with which a JMS trigger polls for more messages.

The **JMS Trigger Management** screen contains the majority of JMS trigger management features. This page, located via **Settings > Messaging > JMS Trigger Management**, displays all of the JMS triggers that exist on the Integration Server along with a summary of each trigger. The summary includes the current status, state, and thread usage of the trigger as well as configuration information such as the JMS connection alias used by the trigger, the destinations to which the trigger subscribes, and the processing mode of the trigger. Integration Server Administrator provides a search capability that you can use to filter the list of displayed triggers or to locate a specific trigger.

Note:

On the **JMS Trigger Management** page, Integration Server Administrator displays a table for standard JMS triggers and another for SOAP-JMS triggers. However, Integration Server Administrator displays the SOAP-JMS triggers table only if a package on Integration Server contains SOAP-JMS triggers and that package is loaded.

The following sections provide more information about managing JMS triggers.

Note:

Unless otherwise specified, the term “JMS triggers” encompasses SOAP-JMS triggers.

Searching for JMS Triggers

As the number of JMS triggers that exist on Integration Server increase, it may become difficult to locate a particular **JMS triggers on the Settings > Messaging > JMS Trigger Management** page in Integration Server Administrator. To assist you with finding JMS triggers on the **JMS Trigger Management** page or to simply filter the list of displayed triggers, Integration Server Administrator includes the ability to search for JMS triggers. You can specify one or more of the following criteria when searching for a JMS trigger:

- JMS trigger name
- JMS connection alias name used by the JMS trigger
- Destinations to which the JMS trigger subscribes

Keep the following information in mind when searching for JMS triggers on the **Settings > Messaging > JMS Trigger Management** page in Integration Server Administrator:

- The search criteria is an AND search. For example, if you specify a JMS trigger name and a JMS connection alias, Integration Server searches for JMS triggers that meet both criteria.
- You can use the wildcard character asterisk (*). You can include a wildcard before and/or after the string used as search criteria. Integration Server does not support the use of any other wildcard characters when searching for JMS triggers.
- The search is case-sensitive.

➤ **To search for JMS triggers**

1. Open the Integration Server Administrator.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under **JMS Configuration**, click **JMS Trigger Management**.
4. Click **Search Triggers**.

Integration Server Administrator displays the Search Criteria fields.

5. Specify search criteria for one or more of the following:

For this criteria **Specify**

Trigger Name The name or a portion of the name of the fully qualified JMS trigger. If you supply a portion of a trigger name, you must include one or more wildcard characters.

Connection Alias Name The name of the JMS connection alias that the JMS trigger uses to retrieve messages from the JMS provider.

Destinations The destinations to which the JMS trigger subscribes.

6. Click **Search**.

Integration Server searches for JMS trigger that meet the provided criteria and then displays the search results on the **JMS Trigger Management** page. You can enter additional search criteria to continue filtering the list of displayed JMS triggers. Click **Show All Triggers** to clear the search results and display all JMS triggers on the Integration Server.

About JMS Trigger Status and State

Integration Server Administrator displays information about the status and state of JMS triggers on the JMS Trigger Management page. The state of a JMS trigger indicates whether the trigger is enabled, disabled, or suspended. The status indicates whether or not the trigger is running.

A JMS trigger can have one of the following states:

Trigger State	Description
Enabled	<p>The JMS trigger is available. A JMS trigger must be enabled for it to receive and process messages.</p> <p>An enabled trigger can have a status of “Not Running” which means that it would not receive and process messages. Reasons that an enabled JMS trigger can be disabled include: a disabled JMS connection alias, an exception thrown by the trigger, and trigger failure at startup.</p>
Disabled	<p>The JMS trigger is not available. Integration Server neither retrieves nor processes messages for the JMS trigger. The JMS trigger remains in this state until you enable the trigger.</p>
Suspended	<p>The JMS trigger is running and connected to the JMS provider. Integration Server has stopped message retrieval, but continues processing any messages it has already retrieved. Integration Server enables the JMS trigger automatically upon server restart or when the package containing the JMS trigger reloads.</p>

When the listed JMS trigger is the trigger group header, the **State** column displays a summary of the state of all the JMS triggers in the trigger group.

- If all triggers in the trigger group have the same state, then the trigger group header will display that state.
- If at least one trigger in the trigger group is enabled, then the trigger group header will display "Enabled [x of n]" where x is the number of enabled triggers and n is the total number of triggers in the trigger group.
- If no triggers are enabled but at least one trigger in the group is Suspended, then the trigger group header will display "Suspended [x of n]" where x is the number of suspended triggers and n is the total number of triggers in the trigger group.

A JMS trigger can have a status of “Running” or “Not Running (*reason*)” where *reason* identifies why the trigger is not running, such as “Not Running (trigger disabled)”.

If a listed JMS trigger is the trigger group header, the **Status** column displays a summary of the status of all the triggers in the trigger group.

- If all of the triggers in the group are running, the **Status** column displays “Running”.
- If at least one trigger in the group is running, the **Status** column displays “Running [x of n]”.
- If none of the triggers in the group are running, the **Status** column displays “Not Running”.

Enabling, Disabling and Suspending a JMS Trigger

You can manage JMS triggers and the amount of resources they consume by changing the state of a JMS trigger.

Using the Integration Server Administrator, you can:

- Enable, disable, or suspend all JMS triggers.
- Enable, disable, or suspend all the JMS triggers of a specific type (standard or SOAP-JMS).
- Enable, disable, or suspend specific JMS triggers.
- Enable, disable, or suspend all of the triggers in a JMS trigger group

You might want to change the state of all JMS triggers at once as a quick way of freeing up server resources. This can be especially helpful in a situation in which Integration Server is functioning under heavy load and additional resources are needed immediately.

You might want to change the state for a specific JMS trigger in the following situations:

- When a back-end system needs maintenance or is becoming unresponsive, you might want to suspend JMS triggers that interact with the back-end system. When the back-end system becomes available, you can enable the JMS triggers.
- After suspending or disabling all JMS triggers, you might enable specific JMS triggers. For example, if the server is functioning under an unusually heavy load, you might first suspend all JMS triggers and then gradually enable JMS triggers, starting with the JMS triggers involved in key processes.
- After Integration Server suspends a serial JMS trigger automatically because a fatal error occurred during message processing. For information about configuring a JMS trigger for fatal error handling, see *webMethods Service Development Help*.
- After creating a WS endpoint trigger by creating a provider web service endpoint alias for JMS.

Important:

If you disable or suspend a SOAP-JMS trigger that acts as a listener for one or more provider web service descriptors, Integration Server will not retrieve any messages for those web service descriptors.

The following procedure explains how to use Integration Server Administrator to change the state of all or individual JMS triggers.

➤ **To enable, disable, or suspend JMS triggers**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **JMS Trigger Management**.
4. If you want to change the state of all JMS triggers, do the following:
 - a. Under Individual JMS Trigger Controls, in the **Enabled** column, click **edit all**.

- b. On the **Settings > Messaging > JMS Trigger Management > Edit Triggers State** screen, in the **New State** list, select the state that you want to apply to all JMS triggers.
- c. In the **JMS Trigger Type** list, select one of the following:

Select	To
Standard	Apply the state change to standard JMS triggers only.
SOAP	Apply the state change to SOAP-JMS triggers only, including WS endpoint triggers.
All	Apply the state change to all JMS triggers.

5. If you want to change the state of a specific JMS trigger, do the following:
 - a. Under Individual JMS Trigger Controls, in the row for the JMS trigger that you want to enable, disable, or suspend, click the text in the **State** column
 - b. On the **Settings > Messaging > JMS Trigger Management > Edit State** screen, in the **New State** list, select the state that you want to apply to this JMS trigger.
6. If you want to change the state for all of the JMS triggers in a JMS trigger group, do the following:
 - a. Under Individual JMS Trigger Controls, in the row for the JMS trigger group header for the trigger group for which you want to enable, disable, or suspend all the triggers, click the text in the **State** column
 - b. On the **Settings > Messaging > JMS Trigger Management > Edit State - JMS Trigger Group** screen, in the **New State** list, select the state that you want to apply to this JMS trigger.
7. Click **Save Changes**.

Notes:

- If you want to disable a JMS trigger, first suspend the JMS trigger and wait for all the processing threads complete. Then, disable the JMS trigger. You can view the number of threads currently used by a JMS trigger on the **Settings > Messaging > JMS Trigger Management** screen.
- When you disable a JMS trigger, Integration Server does the following:
 - If the JMS trigger is waiting before making a retry attempt, Integration Server interrupts processing for the JMS trigger.
 - If the JMS trigger is currently processing messages, Integration Server waits a specified amount of time before forcing the JMS trigger to stop processing messages. If it does not

complete in the allotted time, Integration Server stops the message consumer used to receive messages for the JMS trigger and closes the JMS session. At this point the server thread for the JMS trigger continues to run to completion. However, the JMS trigger will not be able to acknowledge the message when processing completes. If the delivery mode of the message is set to persistent, this can lead to duplicate messages.

The time Integration Server waits between the request to disable the JMS trigger and forcing the trigger to stop is specified by the `watt.server.jms.trigger.stopRequestTimeout` property.

- Because administered objects, like destinations, are configured outside of Integration Server, disabling a JMS trigger has no impact on the subscription.
- If a JMS trigger is processing messages at the time it is suspended, the JMS trigger will complete processing of those messages. The JMS trigger also acknowledges the messages to the JMS provider.
- You can use built-in services to enable, disable, and suspend one or more triggers. For information about the `pub.trigger:disableJMSTriggers`, `pub.trigger:enableJMSTriggers`, and `pub.trigger:suspendJMSTriggers` services, see the *webMethods Integration Server Built-In Services Reference*.

Configuring Integration Server to Enable JMS Triggers Automatically after a Consumer Error

Integration Server disables a JMS trigger if there is a problem in the connection to the JMS provider. Integration Server might also disable a JMS trigger if the message consumer used by the trigger encounters an unexpected error that is not caused by a problem in the connection to the JMS provider. These problems include:

- Network latency between Integration Server and the JMS provider
- A wide area network separating Integration Server and the JMS provider
- The arrival of a large JMS message
- Other network-related issues

You can configure Integration Server to automatically attempt to re-enable a JMS trigger that becomes disabled when the message consumer encounters an error that is not caused by a problem in the connection to the JMS provider. To do this, set the `watt.server.jms.trigger.retryOnConsumerError` to `true`. Set the property to `false` to instruct Integration Server to leave the JMS trigger disabled. The default is `true`.

Controlling Thread Usage for JMS Triggers

You can use Integration Server Administrator to limit the number of server threads that JMS triggers can use. By default, Integration Server can consume up to 100% of the server thread pool for JMS triggers. However, server resources also need to be available to perform other functions.

Viewing Thread Usage for JMS Triggers

Integration Server Administrator displays the number of server threads currently used by each JMS trigger on Integration Server.

➤ To view thread usage for JMS triggers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **JMS Trigger Management**.

Under **Individual Standard JMS Trigger Controls**, in the **Current Threads** column, Integration Server Administrator displays the number of server threads currently used to receive and process messages for each JMS trigger. The **Current Threads** column displays Not Connected if Integration Server is not connected to a JMS provider.

Throttling Thread Usage for JMS Triggers

Integration Server provides controls that you can use to throttle the server thread usage by JMS triggers. You can use the controls to:

- Set the percentage of the server thread pool that Integration Server can use for receiving and processing all JMS triggers.
- Reduce maximum execution threads by the same percentage across all concurrent JMS triggers. This also decreases the rate at which concurrent JMS triggers process messages.

➤ To throttle thread usage for JMS triggers

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **JMS Trigger Management**.
4. Click **Edit JMS Global Trigger Controls**.
5. In the **Thread Pool Throttle** field, type the maximum percentage of the server thread pool that can be used for JMS triggers. This includes threads used to retrieve messages from the JMS provider and threads used to process messages. You must enter a value greater than zero.
6. In the **Individual Trigger Processing Throttle** field, select the value, as a percentage of the configured maximum execution threads value, at which you want to the server to function.

Integration Server applies this percentage to the maximum execution threads value for all concurrent JMS triggers.

This value applies to concurrent JMS triggers only.

7. Click **Save Changes**.

Notes:

- If the **Thread Pool Throttle** percentage does not evaluate to a whole number when applied to the size of the server thread pool, Integration Server rounds up or down to the nearest whole number.
- Serial JMS triggers always process one message at a time. For a serial trigger, Integration Server uses the same thread for receiving and processing a message. The **Individual Trigger Processing Throttle** does not affect serial JMS triggers.
- Concurrent JMS triggers use a pool of consumers to receive and process messages. Each consumer will use a thread from the server thread pool to receive and process a message. A concurrent JMS trigger dedicates an additional thread per connection to managing the pool of consumers. For example, a concurrent JMS trigger configured to use one connection to the JMS provider and configured to process up to 10 messages at a time can use a maximum of 11 server threads. As another example, a concurrent JMS trigger configured to use 2 connections to the JMS provider and configured to process up to 10 messages at a time can use a maximum of 12 threads. For more information about using multiple connections for a concurrent JMS trigger, see [“Allowing Multiple Connections for a JMS Connection Alias” on page 297](#).
- For a concurrent JMS trigger configured to use multiple connections to the webMethods Broker, the trigger's maximum execution threads will never be less than the number of connections specified in the trigger's **Connection count** property. This is true even when reducing the **Individual Trigger Processing Throttle**.
- If the percentage by which you reduce concurrent JMS trigger execution threads does not resolve to a whole number for a JMS trigger, Integration Server rounds up or rounds down to the nearest whole number. However, if rounding down would set the value to 0, the Integration Server rounds up to 1. For example, if you reduce **Individual Trigger Processing Throttle** to 10% of maximum, a concurrent JMS trigger with a maximum execution threads value of 12 would have an adjusted value of 1 (Integration Server rounds 1.2 down to 1). A concurrent JMS trigger with a maximum execution threads value of 4 would have an adjusted value of 1 (Integration Server rounds 0.4 up to 1).
- When you reduce the **Individual Trigger Processing Throttle** and save your changes, Integration Server does not meet the adjusted maximum by terminating any threads currently executing concurrent JMS triggers. Integration Server allows server threads processing messages for concurrent JMS triggers to execute to completion. Integration Server waits until the number of threads executing for a concurrent JMS trigger is less than the adjusted maximum before dispatching another server thread for that JMS trigger.

Establishing a Threshold for Using Multiple Threads with a Concurrent JMS Trigger

By default, concurrent JMS triggers begin using multiple server threads when there is more than one message to process. Integration Server increases the number of server threads used by the trigger until the JMS trigger uses the maximum number of threads that it can to retrieve and process messages. The maximum number of threads that a JMS trigger can use is determined by the **Max execution threads** property value set for the JMS trigger.

To limit the number of threads used by concurrent JMS triggers, you can establish a threshold that determines when the JMS trigger becomes multithreaded. A threshold can be beneficial when a concurrent JMS trigger requires additional processing resources at specific times during the day due to an influx of messages, but for the majority of the day the JMS trigger receives relatively few messages. When the rate of incoming messages is low, the JMS trigger may need only a single thread to process messages efficiently.

The threshold is based on the consecutive number of successful requests for more messages that the JMS trigger makes to the JMS provider. When a threshold is in use, a concurrent JMS trigger initially uses a single thread to retrieve and process messages. When the number of successful consecutive requests for more messages is equal to the specified threshold, the JMS trigger becomes multithreaded. The maximum number of threads Integration Server can use for the JMS trigger is equal to the value of the **Max execution threads** property plus one. (When a concurrent JMS trigger is multithreaded, Integration Server dedicates a server thread to managing the message consumer pool for the trigger.) The concurrent JMS trigger remains multithreaded until the message consumers created by the threads time out. The `watt.server.jms.trigger.pooledConsumer.timeout` property value determines when a message consumer times out. When a consumer times out, the JMS trigger releases the associated server thread. Eventually, when no messages are being received, the concurrent JMS trigger returns to using a single thread.

To use a threshold to determine when a JMS trigger transitions from being single threaded to multithreaded, set the `watt.server.jms.trigger.concurrent.consecutiveMessageThreshold` server property to a positive integer greater than 0 (zero). This property indicates the consecutive number of successful requests for more messages from the JMS provider that the concurrent JMS trigger must make before the trigger becomes multithreaded. For example, when this property is set to 1000, a concurrent JMS trigger becomes multithreaded it makes 1000 consecutive successful requests for more messages from the JMS provider. The default for this property is 0, which indicates that the use of a threshold for making a concurrent JMS trigger multithreaded is disabled. When use of a threshold is disabled, the concurrent JMS trigger will always be multithreaded when more than one message is available for processing.

Note:

The threshold for determining when a concurrent JMS trigger becomes multithreaded applies to all of the concurrent JMS triggers on the Integration Server. This includes standard JMS triggers, SOAP-JMS triggers, and WS-Endpoint triggers. It is not configurable on a per JMS trigger basis.

Configuring Integration Server Session Reuse

You can improve system performance by reusing the same Integration Server session each time a JMS trigger invokes a service. When you reuse Integration Server sessions, you reduce amount of overhead Integration Server requires for each service invocation. Integration Server supports the re-use of Integration Server sessions for both transacted and non-transacted JMS triggers and SOAP-JMS triggers.

You can set JMS triggers to reuse the same Integration Server session for each service invocation by setting the `watt.server.jms.trigger.reuseSession` server configuration parameter to true. By default, this parameter is set to false. For more information about `watt.server.jms.trigger.reuseSession`, see [“Server Configuration Parameters” on page 1017](#).

Each service invocation on the reused session uses the same session ID. You cannot reuse sessions with services that require a unique session ID.

Note:

If you use the Adapter for SAP set `watt.server.jms.trigger.reuseSession` to false if a concurrent JMS trigger has a trigger service that executes multi-step SAP transactions that make calls to the `pub.sap.client.lockSession` and `pub.sap.client.releaseSession` services. These services require unique sessions.

Configuring JMS Session Reuse

You can reduce overhead on JMS providers by reusing JMS sessions across multiple transactions. Non-transacted JMS and SOAP-JMS triggers reuse JMS sessions automatically after receiving and processing messages. To configure this same behavior with transacted JMS triggers and SOAP-JMS triggers, set the `watt.server.jms.trigger.reuseJmsTxSession` server configuration parameter to true. If your JMS provider does not allow the reuse of transacted JMS sessions, set this parameter to false. By default this parameter is set to true. For more information about the `watt.server.jms.trigger.reuseJmsTxSession` server configuration parameter, see [“Server Configuration Parameters” on page 1017](#).

Delaying Polling Requests for Concurrent JMS Triggers

You can reduce the resources that a concurrent JMS trigger uses for message polling by delaying polling requests when the JMS trigger is inactive. If the polling delay feature is enabled, an inactive concurrent JMS trigger waits the specified length of the polling delay before polling the JMS provider for more messages. You can configure Integration Server to gradually increase the delay between polling requests while the JMS trigger remains inactive.

Note:

A concurrent JMS trigger is considered to be inactive if none of the message consumers associated with the JMS trigger received messages in the last round of requests to the JMS provider.

During a polling delay, the JMS trigger will not receive any messages. However, the JMS trigger does not use resources from the JMS provider during the delay. If requests for more messages are expensive for your chosen JMS provider, you might want to use a polling delay to decrease the demands made by an inactive concurrent JMS trigger on the JMS provider.

Configuring Integration Server to delay polling requests has minimal impact on the time required to shut down Integration Server. As part of shut down, Integration Server shuts down each JMS trigger. Integration Server can interrupt the extended delay to begin shutting down the JMS trigger.

To use a polling delay, you need to specify the following:

- The length of the delay that should elapse before the polling interval begins. The `watt.server.jms.trigger.extendedDelay.delays` parameter determines the length of the delay. If you want to use a gradually increasing delay, you indicate the incremental delays by specifying a comma-separated list of integers. The `watt.server.jms.trigger.extendedDelay.delays` parameter is measured in milliseconds and has a default value of: 0, 1000, 2000, 3000
- The time interval at which Integration Server introduces the polling delay for an inactive concurrent JMS trigger. If the JMS trigger remains inactive, the same time interval determines when Integration Server increases the delay between polling requests. The time interval is determined by the `watt.server.jms.trigger.extendedDelay.delayIncrementInterval` parameter. This parameter is measured in millisecond and has a default value or 0 which indicates that Integration Server does not introduce a delay between polling requests for inactive concurrent JMS triggers.

How JMS Trigger Delays a Polling Request

The following table explains how a concurrent JMS trigger polls for messages when Integration Server is configured to delay polling requests for inactive concurrent JMS triggers.

Stage	Description
1	<p>The concurrent JMS trigger becomes inactive.</p> <p>Integration Server considers a concurrent JMS trigger to be inactive if none of the message consumers associated with the JMS trigger received messages in the last round of requests to the JMS provider.</p>
2	<p>If the <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code> parameter is greater than or equal to 1, the polling delay feature is enabled. The JMS trigger begins to delay polling requests by the amount of time specified by the first integer in the <code>watt.server.jms.trigger.extendedDelay.delays</code> parameter.</p> <p>For example, suppose that <code>watt.server.jms.trigger.extendedDelay.delays = 1000, 2000, 3000</code>. When the JMS trigger becomes inactive, the JMS trigger</p> <ol style="list-style-type: none">1. Waits 1000 milliseconds2. Polls the JMS provider for more messages <p>When the <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code> parameter is set to 0, the polling delay feature is disabled. Integration Server will not introduce a delay between polling requests.</p>

Stage	Description
3	<p>After the interval specified by <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code> elapses, the JMS trigger begins to delay polling requests by the amount of time specified by the second integer in <code>watt.server.jms.trigger.extendedDelay.delays</code>.</p> <p>For example, suppose that <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code>, which is measured in milliseconds, is set to 5000. The JMS trigger polls for new messages at the frequency described in Stage 2 for 5000 milliseconds. After 5000 milliseconds elapse, the JMS trigger begins using the second extended delay value. If <code>watt.server.jms.trigger.extendedDelay.delays = 1000, 2000, 3000</code>, the JMS trigger now waits 2000 millisecond before polling the JMS provider for more messages.</p>
4	<p>After the interval specified by <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code> elapses, the JMS trigger begins to delay polling request by the amount of time specified by the third integer in <code>watt.server.jms.trigger.extendedDelay.delays</code>.</p> <p>For example, suppose that <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code>, which is measured in milliseconds, is set to 5000. The JMS trigger polls for new messages at the frequency described in Stage 3 for 5000 milliseconds. After 5000 milliseconds elapse, the JMS trigger begins using the third extended delay value. If <code>watt.server.jms.trigger.extendedDelay.delays = 1000, 2000, 3000</code>, the JMS trigger now waits 3000 milliseconds before polling the JMS provider for more messages.</p>
5	<p>Each time the interval specified by <code>watt.server.jms.trigger.extendedDelay.delayIncrementInterval</code> elapses, the JMS trigger uses the next value in <code>watt.server.jms.trigger.extendedDelay.delays</code> to change the length of time between polling requests. When the JMS trigger reaches the last value in <code>watt.server.jms.trigger.extendedDelay.delays</code>, the JMS trigger uses the specified polling delay until the JMS trigger retrieves a message from the JMS provider.</p>
6	<p>When a polling request to the JMS provide returns one or more messages, Integration Server considers the JMS trigger to be active and stops using a polling delay.</p>

Examples of Extended Polling Delay Configuration

To further illustrate how you might configure an extended polling delay for concurrent JMS triggers, consider the following examples.

Example 1

Suppose that the server configuration parameters for an extended polling delay and concurrent JMS trigger polling interval are set as follows. Keep in mind that the parameters are measured in milliseconds.

- `watt.server.jms.trigger.extendedDelay.delays = 100, 200, 1000`
- `watt.server.jms.trigger.extendedDelay.delayIncrementInterval = 5000`

As soon as the JMS trigger becomes inactive, the JMS trigger waits the polling delay of 100 milliseconds and then polls the JMS provider for more messages. The JMS trigger continues polling, waiting the 100 milliseconds specified by the extended delay between polling requests. If 5000 milliseconds elapse and the JMS trigger remains inactive, the JMS trigger begins using a 200 millisecond delay between polling requests. After 5000 more milliseconds, if the JMS trigger remains inactive, the JMS trigger begins to use an extended delay of 1000 milliseconds. The JMS trigger continues to use the 1000 millisecond extended delay until a polling request returns one or more messages.

Example 2

When Integration Server is configured to use an extended polling delay for concurrent JMS triggers, the first delay takes place as soon as the JMS trigger becomes inactive. If you want the JMS trigger to use an extended delay only after the interval specified by `watt.server.jms.trigger.extendedDelay.delayIncrementInterval` elapses, use 0 as the first value in `watt.server.jms.trigger.extendedDelay.delays`.

Suppose that the server configuration parameters for an extended polling delay and concurrent JMS trigger polling interval are set as follows. Keep in mind that the parameters are measured in milliseconds.

- `watt.server.jms.trigger.extendedDelay.delays = 0, 10000`
- `watt.server.jms.trigger.extendedDelay.delayIncrementInterval = 3600000`

After the JMS trigger has been inactive for an hour (3600000 milliseconds), the JMS trigger waits the polling delay of 10000 milliseconds and then polls the JMS provider for more messages. If the polling request does not return any messages, the JMS trigger waits 10000 milliseconds and then polls the JMS provider for more messages. The JMS trigger continues polling in this way until a polling request returns a message.

What Happens When JMS Triggers Fail to Start?

When a JMS connection alias starts, Integration Server attempts to start all of the JMS triggers that use that JMS connection alias to retrieve messages from the JMS provider. Starting a JMS trigger can fail due to the following reasons:

- The JMS trigger and/or the JMS provider are not configured correctly (for example, the lookup name for the destination that the trigger subscribes to does not exist on the JNDI provider or the Integration Server does not have sufficient privileges to retrieve messages from the destination). If incorrect configuration of the JMS trigger or JMS provider prevents JMS triggers from starting, Integration Server can take no further action to start the trigger. Integration Server logs an exception which will be visible in the JMS Trigger Management page of Integration Server Administrator. The JMS trigger remains inactive until the configuration is resolved and the trigger is manually restarted (i.e. enabled).
- The JMS provider is not available at the time Integration Server attempts to start the JMS trigger. If the JMS provider is not available, the JMS connection alias will fail as well. When a connection failure causes a JMS connection alias to stop Integration Server attempts to restart the JMS connection alias every 20 seconds. When the connection to the JMS provider is restored, the JMS connection alias restarts, and Integration Server attempts to start all of the JMS triggers that use the JMS connection alias.
- The destination is temporarily unavailable on the JMS provider. In the case of Universal Messaging, a queue/channel may be unavailable temporarily if the user edits or performs maintenances on the queue/channel at the same time Integration Server attempts to start the JMS trigger. To handle this temporary situation, Integration Server retries starting the trigger if `watt.server.jms.trigger.startupFailure.retryCount` server configuration parameter is set to a value greater than 0 and one of the following is true:
 - The JMS trigger is configured to use more than one connection and at least one of the trigger instances has already started successfully. When a JMS trigger uses multiple connections, a new instance of the trigger will be started for each connection.
 - The JMS trigger failed because of a `javax.jms.InvalidDestinationException`. Note that a `javax.jms.InvalidDestinationException` does not include a JNDI lookup failure which is a `javax.naming.NamingException`.

The `watt.server.jms.trigger.startupFailure.retryCount` parameter value determines the maximum number of retry attempts that Integration Server makes to start a JMS trigger after the trigger fails to start. After the initial trigger startup failure, Integration Server waits an interval of 1000 milliseconds and then retries starting the trigger. If startup of the trigger fails, Integration Server waits 1000 milliseconds and then makes another attempt to start the trigger. Integration Server continues in this way until the JMS trigger starts successfully, the JMS connection alias stops running, or the JMS trigger becomes disabled. When starting the trigger fails after Integration Server makes the maximum number of retry attempts, Integration Server can take no further action to start the trigger. Integration Server logs an exception which will be visible in the JMS Trigger Management page of Integration Server Administrator. The JMS trigger remains inactive until the problem is resolved and the trigger is manually restarted (i.e. enabled).

The `watt.server.jms.trigger.startupFailure.retryCount` parameter must be set to an integer greater than or equal to 0. When the parameter is set to 0, the default, Integration Server does not make any retry attempts.

Excessive retry attempts may cause the JMS connection alias startup time to be delayed because Integration Server uses the same thread to make retry attempts that it uses to start the alias.

It is possible that a correctly configured JMS trigger fails to start after the JMS connection alias starts up successfully. This is sometimes observed when the JMS connection alias connects to one

Universal Messaging server in a Universal Messaging cluster and the JMS trigger connects to a different server in the cluster. In some cases, the JMS trigger may start successfully after a short delay. To allow these triggers to start up automatically, Integration Server schedules a task to retry starting the JMS trigger. For more information about the trigger restart task, see [“How Does the JMS Trigger Restart Task Work?”](#) on page 874.

How Does the JMS Trigger Restart Task Work?

If a JMS trigger does not start when a JMS connection alias starts, Integration Server schedules a task to retry starting the JMS triggers and then continues with activities associated with starting the JMS connection alias. The trigger restart task, which runs in its own thread, attempts to restart the failed JMS triggers at a set interval. By default, the task executes every 30 seconds. The task completes after all failed triggers for the JMS connection alias successfully start, when the JMS connection alias is disabled, or the maximum retry attempts have been made. If the maximum retry attempts have been made and JMS triggers still do not start, the remaining JMS triggers likely need to be examined for configuration problems. When the configuration issues are resolved, the JMS trigger can be restarted manually using Integration Server Administrator.

Integration Server schedules one task for each JMS connection alias that has triggers that do not start when the alias starts up successfully. Integration Server writes server log messages as the Info, Debug, and Trace levels to indicate the progress of the triggers restart task. Because the restart task uses a separate thread from the one used to start the JMS connection alias, attempts to restart the JMS trigger do not impede the completion of the connection alias start up.

Integration Server provides server configuration parameters that you can use to control the interval at which the trigger restart task attempts to start the JMS triggers and the interval the task waits between attempts.

- `watt.server.jms.trigger.startupFailure.restartTaskRetryCount`. Specifies the maximum number of retry attempts the trigger restart task makes to start the JMS triggers that fail to start when the JMS connection alias starts up. Integration Server makes the initial start attempt for a JMS trigger when starting the JMS connection alias used by the trigger. The default is 6 retry attempts.
- `watt.server.jms.trigger.startupFailure.restartTaskRetryInterval`. Specifies the number of seconds that the trigger restart task waits between attempts to restart JMS triggers that failed to start when the JMS connection alias started. You must specify a positive integer. The default is 30 seconds.

Once the JMS trigger starts and is running, it is possible for the JMS trigger to catch an exception while receiving messages from the JMS provider. Typically, this occurs when the JMS provider becomes unavailable. If the JMS provider is not available, Integration Server disables the JMS trigger. The JMS connection alias will eventually fail as well. When a connection failure causes a JMS connection alias to stop Integration Server attempts to restart the JMS connection alias every 20 seconds. When the connection to the JMS provider is restored, the JMS connection alias restarts, and Integration Server attempts to start all of the JMS triggers that use the JMS connection alias.

It is possible that a JMS trigger will become disabled but the JMS connection alias recovers quickly and does not stop. Because the JMS connection alias does not fail because of a connection error, Integration Server does not attempt to restart the JMS connection alias, which means that Integration

Server does not make the automatic attempt to restart the JMS triggers after the alias restarts. Here, if `watt.server.jms.trigger.retryOnConsumerError` is set to true, Integration Server attempts to start JMS triggers automatically. Integration Server attempts indefinitely to start the JMS trigger as long as the JMS connection alias is running and the JMS trigger is enabled.

About WS Endpoint Triggers

A WS (Web Service) endpoint trigger is a SOAP-JMS trigger with limited configuration options. A WS endpoint trigger is associated with a provider web service endpoint alias and acts as a listener for any provider web service descriptors to which the alias is assigned. Typically, WS endpoint triggers are used with virtual services deployed to webMethods API Gateway.

Keep the following points in mind when working with WS endpoint triggers:

- Integration Server uses the following naming convention when creating a WS endpoint trigger: WS Endpoint Trigger: *aliasName* where *aliasName* is the provider web service endpoint alias name.
- Integration Server saves WS endpoint triggers to the following configuration file:
Integration Server_directory /instances/*instance_name*/config/endpoints/providerJMS.cnf.
- The existence of the WS endpoint trigger is tied to the provider web service endpoint alias. If you select **WS Endpoint Trigger** as the JMS trigger to use with a provider web service endpoint alias, Integration Server creates the WS endpoint trigger a part of creating the alias. If you delete the alias, Integration Server deletes the WS endpoint trigger as well.
- At the time it creates the WS endpoint trigger, Integration Server sets some default properties, such as serial processing and the JMS connection alias. You can use Integration Server Administrator to change the default values.

Note:

While WS endpoint triggers are SOAP-JMS triggers, WS endpoint triggers have a subset of the properties and features of a SOAP-JMS trigger. For example, exactly-once processing, fatal error handling, and transient error handling are available for SOAP-JMS triggers only.

- When Integration Server creates a WS endpoint trigger, Integration Server assigns a placeholder destination. The destination name uses the format: `wseQueue_aliasName` where *aliasName* is the provider web service endpoint alias name. Unless a queue with this name exists, you must specify the actual destination to which the WS endpoint trigger subscribes.
- When Integration Server creates a WS endpoint trigger, Integration Server leaves the WS endpoint trigger in a disabled state. You must enable the trigger for it to receive messages from the JMS provider. For more information about enabling JMS triggers, see [“About JMS Trigger Status and State” on page 861](#).
- You can only edit the WS endpoint trigger using Integration Server Administrator. You cannot use Designer to edit a WS endpoint trigger.
- Integration Server sets the execution user to Administrator for all WS endpoint triggers.

Editing WS Endpoint Triggers

When Integration Server creates a WS endpoint trigger as part of creating a provider web service endpoint alias, Integration Server uses default values for some of the trigger properties and placeholders for other properties. You can use Integration Server Administrator to change the default and placeholder values.

Keep the following guidelines in mind when editing a WS endpoint trigger:

- The JMS connection alias you want Integration Server to use to obtain connections to and receive messages from the JMS provider must already exist. Although a JMS connection alias does not need to be enabled at the time you assign it to the WS endpoint trigger, the JMS connection alias must be enabled for the WS endpoint trigger to execute at run time.
- The transaction type of the JMS connection alias determines whether or not the WS endpoint trigger is transacted (that is, it receives and processes messages as part of a transaction)
- The destination to which the WS endpoint trigger subscribes must be a queue.
- Only concurrent triggers can have a **Max Execution Threads** value greater than one.
- To specify a **Connection Count** value greater than one, the following must be true:
 - The specified JMS connection alias must be configured to create a new connection for each JMS trigger that uses that connection alias. For more information, see [“Allowing Multiple Connections for a JMS Connection Alias”](#) on page 297.
 - The WS endpoint trigger must be configured for concurrent processing.

➤ To edit a WS endpoint trigger

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **JMS Trigger Management**.
4. Under **Individual SOAP JMS Trigger Controls**, select the WS endpoint trigger that you want to edit.
5. On the **Settings > Messaging > JMS Trigger Management > Detail > triggerName** screen, click **Edit Trigger**.
6. In the **JMS Connection Alias** list, select the JMS connection alias that you want the WS endpoint trigger to use to receive messages from the JMS provider.
7. In the **Destination Name** field, type the name of the queue from which the WS endpoint trigger will receive messages.

8. Next to **Processing Mode**, select one of the following:

Select	To
Serial	Specify that Integration Server processes messages received by the trigger one after the other.
Concurrent	Specify that Integration Server processes multiple messages for this trigger at one time.

9. If you selected concurrent processing, in the **Max execution threads** field, specify the maximum number of messages that Integration Server can process concurrently.
10. If you selected concurrent processing, in the **Connection Count** field, specify the number of connections you want the WS endpoint trigger to make to the webMethods Broker.

Note:

The **Connection Count** value must be less than or equal to the **Max Execution Threads** value.

11. Click **Save Changes**.

46 Managing MQTT Triggers

■ Introduction to MQTT Trigger Management	880
■ MQTT Trigger State and Status	880
■ Configuring Integration Server Session Reuse for MQTT Triggers	882
■ Automatic Retry for Starting MQTT Triggers	883

Introduction to MQTT Trigger Management

Integration Server provides ways to manage MQTT triggers and the resources that the triggers consume, including:

- Changing the trigger state. This determines whether the MQTT trigger receive and/or processes messages.
- Configuring Integration Server to reuse sessions for MQTT triggers.
- Configuring Integration Server to automatically attempt to start or MQTT triggers that fail to start initially.

MQTT Trigger State and Status

Each MQTT trigger has a state and a status. The state indicates whether the trigger has been set to enabled, disabled, or suspended. The status reflects the current condition of the trigger. An MQTT trigger can have a status of “Running” or “Not Running (*reason*)” where *reason* identifies why the trigger is not running, such as “Not Running (Trigger Disabled)” or “Not running (Connection Alias Failure)” For example, an MQTT trigger can have a state of Enabled and a status of Not running (Trigger Failure).

The following table describes the possible states for an MQTT trigger.

Trigger State	Description
Enabled	The MQTT trigger is running and connected to the MQTT server. Integration Server retrieves and processes messages for the MQTT trigger.
Disabled	<p>The MQTT trigger is stopped. The MQTT trigger is not connected to the MQTT server. Integration Server neither retrieves nor processes messages for the MQTT trigger. The MQTT trigger remains in this state until you enable it.</p> <p>An MQTT trigger can be in a temporary disabled state which is indicated by an asterisk * after the word Disabled. An MQTT trigger can become temporarily disabled when the Stop trigger retry failure handling behavior takes effect after retry failure occurs. For more information about retry failure handling and when Integration Server automatically enables a temporarily disabled MQTT trigger, see <i>webMethods Service Development Help</i>.</p>
Suspended	The MQTT trigger is running and connected to the MQTT server but is not retrieving or processing new messages. Integration Server stopped retrieving messages from the MQTT server, stopped pulling messages from the local trigger queue for the MQTT trigger, but continues processing any messages it has already retrieved from the local trigger queue.

Trigger State	Description
	Integration Server enables the MQTT trigger automatically upon server restart or when the package containing the MQTT trigger reloads.

Changing the State of an MQTT Trigger

You can change the state of a single MQTT trigger or all of the MQTT triggers on the Integration Server.

Keep the following information in mind when changing the state of an MQTT trigger:

- If you want to disable an MQTT trigger, first suspend the MQTT trigger and wait for all trigger processing threads to complete. Then disable the trigger. You can view the number of threads currently used by a MQTT trigger in the **Current Threads** column on the **Settings > Messaging > MQTT Trigger Management** page.
- If you disable an MQTT trigger without first suspending it, message processing might not complete before the trigger is disabled.
- When you suspend an enabled MQTT trigger, Integration Server:
 - Stops retrieving messages from the MQTT server.
 - Stops pulling messages for processing from the local trigger queue.
 - Continues to process any messages that the trigger already retrieved from the local trigger queue.
- When you disable a suspended MQTT trigger for which all message processing has completed, Integration Server:
 - Clears the local trigger queue.
 - Stops the listener that the trigger uses to retrieve messages. This disconnects the MQTT trigger from the MQTT server.
- When you disable an enabled MQTT trigger, (that is, you do not first suspend the trigger), Integration Server:
 - Stops retrieving messages from the MQTT server.
 - Stops pulling messages for processing from the local trigger queue.
 - Waits a short amount of time for message processing to complete. If message processing has not completed by the time the timeout elapses, Integration Server stops the listener for the MQTT trigger. After Integration Server stops the listener Integration Server cannot acknowledge any outstanding messages Any server thread processing message for the MQTT trigger continues to completion; however, acknowledgment of the MQTT message upon completion of processing will fail.

The server configuration parameter `watt.server.commonmessaging.trigger.stopRequestTimeOut` determines the maximum time that Integration Server waits for processing to complete. The default is 3 seconds.

- If the MQTT trigger is waiting before making a retry attempt due to transient error handling, Integration Server interrupts processing for the MQTT trigger. No further retry attempts will be made.
- Clears the local trigger queue.
- Stops the listener that the trigger uses to retrieve messages. This disconnects the MQTT trigger from the MQTT server.

➤ **To change the state of an MQTT trigger**

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Click **MQTT Trigger Management**.
4. If you want to change the state of all MQTT triggers, do the following:
 - a. In the **Enabled** column, click **edit all**.
 - b. On the **Settings > Messaging > MQTT Trigger Management > Edit Triggers State** screen, in the **New State** list, select the state that you want to apply to all MQTT triggers.
5. If you want to change the state of a specific MQTT trigger, do the following:
 - a. In the row for the MQTT trigger that you want to enable, disable, or suspend, select the current text in the **State** column
 - b. On the **Settings > Messaging > MQTT Trigger Management > Edit State** screen, in the **New State** list, select the state that you want to apply to this MQTT trigger.
6. Click **Save Changes**.

Configuring Integration Server Session Reuse for MQTT Triggers

You can improve server performance by reusing the same Integration Server session each time an MQTT trigger invokes a service. When you reuse sessions, you reduce the amount of overhead Integration Server requires for a service invocation.

Each service invocation with the reused session uses the same session ID. You cannot reuse sessions with services that require a unique session ID.

You can configure all MQTT triggers to reuse Integration Server sessions by setting the `watt.server.commonmessaging.trigger.reuseSession` server configuration parameter to true. By default, this parameter is set to false.

Automatic Retry for Starting MQTT Triggers

As part of starting or an MQTT connection alias, Integration Server also starts the enabled MQTT triggers that use the alias to retrieve messages. If an MQTT connection alias fails or becomes unavailable after being successfully started, Integration Server attempts to restart the enable MQTT triggers once the connection alias is successfully restarted. During the start or restart process for the alias, attempts to start (or restart) the MQTT triggers might fail. For example, if the MQTT trigger and/or MQTT server are not properly configured, the trigger will not start.

If an MQTT trigger does not start when an MQTT connection alias starts (or restarts), Integration Server schedules a task to retry starting the MQTT triggers and then continues with activities associated with starting the MQTT connection alias. The trigger restart task, which runs in its own thread, attempts to restart the failed MQTT triggers at a set interval. By default, the task executes every 30 seconds. The task completes after all failed triggers for the MQTT connection alias successfully start, when the MQTT connection alias is disabled, or the maximum retry attempts have been made. If the maximum retry attempts have been made and MQTT triggers still do not start, the remaining MQTT triggers likely need to be examined for configuration problems. When the configuration issues are resolved, the MQTT trigger can be restarted manually using Integration Server.

Integration Server schedules one task for each MQTT connection alias that has triggers that do not start when the alias starts up successfully. Integration Server writes server log messages as the Info, Debug, and Trace levels to indicate the progress of the triggers restart task. Because the restart task uses a separate thread from the one used to start the MQTT connection alias, attempts to restart the MQTT trigger do not impede the completion of the connection alias start up.

Integration Server provides server configuration parameters that you can use to control the interval at which the trigger restart task attempts to start the MQTT triggers and the interval the task waits between attempts.

- `watt.server.commonmessaging.trigger.restartTaskRetryCount`. Specifies the maximum number of retry attempts the trigger restart task makes to start the MQTT triggers that fail to start when the MQTT connection alias starts up. Integration Server makes the initial start attempt for an MQTT trigger when starting the MQTT connection alias used by the trigger. The default is 6 retry attempts.
- `watt.server.commonmessaging.trigger.restartTaskRetryInterval`. Specifies the number of seconds that the trigger restart task waits between attempts to restart MQTT triggers that failed to start when the MQTT connection alias started. You must specify a positive integer. The default is 30 seconds.

47 Using a Document History Database with Exactly-Once Processing

■ Overview	886
■ Removing Expired Entries from the Document History Database	886
■ Viewing Exactly-Once Processing Statistics	887
■ Clearing Exactly-Once Processing Statistics	887

Overview

The document history database maintains a record of guaranteed documents processed by webMethods messaging triggers and persistent messages processed by JMS triggers. The database keeps a document history only for triggers that specify that document history should be used as part of duplicate detection. Integration Server adds entries to the document history database when a trigger service begins executing and when it executes to completion (whether it ends in success or failure).

For more information about exactly-once processing for webMethods messaging triggers or JMS triggers, see *webMethods Service Development Help*.

Removing Expired Entries from the Document History Database

To keep the size of the document history database manageable, Integration Server periodically removes expired rows from the database. The value of the trigger's **History time to live** property determines how long the document history database maintains an entry for a processed document or message.

Integration Server provides a scheduled service to remove expired entries from the database. By default, the Message History Sweeper task executes every 10 minutes. You can change the frequency with which the service executes. For information about editing scheduled services, see [“Scheduling Services” on page 715](#).

Note:

The `watt.server.idr.reaperInterval` property determines the initial execution frequency for the Message History Sweeper task. After you define a JDBC connection pool for Integration Server to use to communicate with the document history database, change the execution interval by editing the scheduled service.

You can also clear all expired entries from the database at any time. For more information, see [“Clearing Expired Entries from the Document History Database” on page 886](#).

Clearing Expired Entries from the Document History Database

If you do not want to wait for the next scheduled execution of the Message History Sweeper task to remove expired entries, you can use Integration Server Administrator to remove all of the expired entries.

➤ To clear expired entries from the document history database

1. Open Integration Server Administrator.
2. From the **Settings** menu in the Navigation panel, click **Resources**.
3. Click **Exactly Once Statistics**.

4. Click **Remove Expired Document History Entries**.

Viewing Exactly-Once Processing Statistics

You can use Integration Server Administrator to view a history of the In Doubt or Duplicate messages received by JMS triggers. Integration Server Administrator displays the name, UUID (universally unique identifier), and status for the Duplicate or In Doubt messages received by triggers for which exactly-once processing is configured.

Integration Server saves exactly-once statistics in memory. When Integration Server restarts, the statistics will be removed from memory.

➤ To view exactly-once processing statistics

1. Start webMethods Integration Server and open the Integration Server Administrator.
2. Under the **Settings** menu in the navigation area, click **Resources**.
3. Click **Exactly-Once Statistics**.

Clearing Exactly-Once Processing Statistics

You can use Integration Server Administrator to delete statistics for In Doubt documents and Duplicate documents.

➤ To clear exactly-once processing statistics

1. Start webMethods Integration Server and open the Integration Server Administrator.
2. Under the **Settings** menu in the navigation area, click **Resources**.
3. Click **Exactly-Once Statistics**.
4. Click **Clear All Duplicate or In Doubt Document Statistics**.

48 Using Integration Server to Manage XA Transactions

■ Overview of XA Transaction Management	890
■ Configuring XA Options in Integration Server	894
■ Manually Resolving a Transaction	897

Overview of XA Transaction Management

A transaction is a logical unit of work, composed of many different processes, that either entirely succeeds or has no effect at all.

Because the work being performed within a transaction can occur on many different platforms and can involve many different resources from different vendors, the X/Open organization developed the distributed transaction process (DTP) model and the XA interface.

The DTP model defines communication among the following:

- *Resources* such as databases.
- *Resource managers* such as database servers. Resource managers provide access to shared resources.
- A *transaction manager*. The transaction manager coordinates and controls all transactions for resources through their resource managers. Integration Server contains a transaction manager sub-system that coordinates and controls database transactions that are initiated by the webMethods Adapter for JDBC or the webMethods Adapter for JMS.

The XA interface describes the protocol for transaction coordination, commitment, and recovery between resource and a transaction manager. Integration Server supports the XA protocol; it manages transactions using the transaction protocol called *two-phase commit*, or 2PC.

In the first phase of 2PC, Integration Server (specifically, the transaction manager) asks the resources that are participating in a transaction whether they are prepared to commit the transaction. In the second phase, one of the following occurs:

- All the resources respond that they are prepared to commit. Integration Server instructs the resources to commit the transaction.
- One or more resources respond that they are not prepared to commit. Integration Server instructs all resources to roll back their preparations for committing the transaction.

How the Integration Server Persists the State of a Transaction

At the beginning of a transaction, Integration Server creates a unique transaction ID called an *XID*. The Integration Server stores the XID and the global state of the transaction in a persistent store called the XA recovery store. At the beginning of each subsequent action taken for the transaction, Integration Server stores the global state of the transaction and the state of each resource that is participating in the transaction in the XA recovery store. If Integration Server ends abnormally, Integration Server can retrieve the state information from the XA recovery store and try to resolve uncompleted transactions, if there are any.

For Integration Server to store state information, the following conditions must be met:

- The `watt.server.transaction.xastore.performXALogging` server configuration parameter must be set to true. Integration Server writes transaction information to the XA recovery store only when this property is set to true, which is the default.

- The transaction involves multiple resources and all the resources are XA-enabled (that is, the resources support the JTA and XA standards and keep persistent records of transactions that have been prepared or heuristically committed).
- The transaction is defined as an XA transaction. For example, if the transaction involves the webMethods Adapter for JDBC, the transaction would be defined as an XA transaction on the adapter's connections to the resources.

Note:

As with most features that improve reliability and recoverability, this feature may increase the overhead associated with processing XA transactions.

How the Integration Server Resolves Uncompleted Transactions

If Integration Server ends abnormally while transactions are in progress, those transactions are uncompleted. When Integration Server restarts, it retrieves a list of uncompleted transactions from the XA recovery store. Based on the last status Integration Server logged for the transactions on the list, Integration Server tries to resolve each transaction, as follows:

If	Integration Server does this
The resources had begun the commit process and at least one resource had committed the transaction	Tries to get the other resources to commit
The resources had finished preparing to commit the transaction but had not begun the commit process	Tells the resources to roll back all preparations for the commit
The resources had begun the commit process but no resource had committed the transaction	Tells the resources to roll back all preparations for the commit
The resources had begun but not completed rolling back the transaction	Tells the resources to roll back all preparations for the commit
Integration Server had not yet asked the resources whether they are prepared to commit the transaction	Forgets the transaction and erases its XID from the XA recovery store
The resources had completed committing or rolling back the transaction	Forgets the transaction and erases its XID from the XA recovery store

If an error occurs while Integration Server is trying to resolve an uncompleted transaction, Integration Server waits a period of time that you specify and then tries again. Integration Server continues trying to resolve the uncompleted transaction until a maximum recovery time that you specify expires. For more information about configuring these values, see [“Configuring XA Server Parameters” on page 896](#).

New XA transactions continue unimpeded during Integration Server's attempts at resolution.

Integration Server cannot resolve all uncompleted transactions. For example, Integration Server cannot resolve a transaction in these cases:

- A resource administrator forced a commit or rollback of a transaction on a resource after Integration Server ended abnormally.
- The transaction includes a 1PC (one-phase commit) resource, and Integration Server stores statuses only for transactions whose participating resources are all XA-enabled.
- Integration Server cannot connect to the resource after repeated attempts within the specified maximum recovery time (for example, because the transaction involves the webMethods Adapter for JDBC and the adapter's connection to the resource does not exist or has been changed).

In such cases, you will have to resolve the uncompleted transaction manually.

About Unresolved XA Transactions

Integration Server displays the XA transactions that need to be manually resolved on the **Settings > Resources > XA Manual Recovery** screen. Integration Server lists the unresolved transaction in a table and displays the following information about each unresolved transaction.

Column	Description																				
XID	Unique XID for the transaction. Integration Server created the XID at the beginning of the transaction and wrote it to the XA recovery store; Integration Server also provided the XID to the participating resources, which also stored the information.																				
Global 2PC State	Global state of the transaction before Integration Server ended. If a state maps to a global state in the <code>javax.transaction.Status</code> interface described in the JTA standard, that mapping is shown below.																				
	<table border="1"> <thead> <tr> <th>State</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>TR_PREPARE_BEGIN</td> <td>STATUS_PREPARING</td> </tr> <tr> <td>TR_PREPARE_RESOURCE</td> <td></td> </tr> <tr> <td>TR_PREPARE_RESOURCE_END</td> <td></td> </tr> <tr> <td>TR_PREPARE_END</td> <td>STATUS_PREPARED</td> </tr> <tr> <td>TR_COMMIT_BEGIN</td> <td>STATUS_COMMITTING</td> </tr> <tr> <td>TR_COMMIT_RESOURCE</td> <td></td> </tr> <tr> <td>TR_COMMIT_RESOURCE_END</td> <td></td> </tr> <tr> <td>TR_ROLLBACK_BEGIN</td> <td>STATUS_ROLLING_BACK</td> </tr> <tr> <td>TR_ROLLBACK_RESOURCE</td> <td></td> </tr> </tbody> </table>	State	Description	TR_PREPARE_BEGIN	STATUS_PREPARING	TR_PREPARE_RESOURCE		TR_PREPARE_RESOURCE_END		TR_PREPARE_END	STATUS_PREPARED	TR_COMMIT_BEGIN	STATUS_COMMITTING	TR_COMMIT_RESOURCE		TR_COMMIT_RESOURCE_END		TR_ROLLBACK_BEGIN	STATUS_ROLLING_BACK	TR_ROLLBACK_RESOURCE	
State	Description																				
TR_PREPARE_BEGIN	STATUS_PREPARING																				
TR_PREPARE_RESOURCE																					
TR_PREPARE_RESOURCE_END																					
TR_PREPARE_END	STATUS_PREPARED																				
TR_COMMIT_BEGIN	STATUS_COMMITTING																				
TR_COMMIT_RESOURCE																					
TR_COMMIT_RESOURCE_END																					
TR_ROLLBACK_BEGIN	STATUS_ROLLING_BACK																				
TR_ROLLBACK_RESOURCE																					

Column	Description
	TR_ROLLBACK_RESOURCE_END
	TR_ROLLBACK_END STATUS_ROLLED_BACK
	TR_ROLLBACK_ONLY MARKED_ROLLBACK
	TR_FORGET_RESOURCE
	TR_FORGET_RESOURCE_END
	TR_COMPLETED
	TR_RECOVERY Integration Server is trying to resolve the transaction.
	TR_UNDEFINED STATUS_UNKNOWN
Error Message	Error message that Integration Server wrote to the server log before storing the global state of the transaction in the XA recovery store.
Recovery Action Attempted	Action that Integration Server took to try to resolve the transaction. If Global 2PC State is TR_COMMIT_BEGIN, Integration Server tried to commit the transaction. If the global state is any other value, Integration Server tried to roll back the transaction.

Note:

Refresh the page at intervals to make sure all uncompleted transactions are listed. Suppose Integration Server tries to resolve an uncompleted transaction but cannot; the transaction will not be listed while Integration Server is trying to resolve it, but if you refresh the page later, the transaction will appear on the list.

Details for an Unresolved XA Transaction

For each unresolved XA transaction, you can view detailed information, such as the participating resources, the state of the transaction on each resource, and the inferred status of the transaction on the resource. When you click the XID for an unresolved transaction on the **Settings > Resources > XA Manual Recovery** screen, the Integration Server Administrator displays the following information for each resource involved in the transaction:

Column	Description
XA Resource	Fully qualified name of the resource.
XID Exists?	Indicates whether the transaction's XID exists on the resource.
	<u>This...</u> <u>Indicates...</u>
Yes	That the XID exists on the resource.

Column	Description															
No	That the XID does not exist.															
Unknown	That the Integration Server could not determine whether the XID exists on the resource.															
State	Current state of the transaction on the resource. The values are the same as those in the Global 2PC State list. For a list of global 2PC states, see the table in “About Unresolved XA Transactions” on page 892.															
Inferred Status	Assumed status of the transaction on the resource, based on the values of XID exists and State . Based on the possible combinations, statuses are as follows:															
	<table border="1"> <thead> <tr> <th>XID Exists?</th> <th>State</th> <th>Inferred Status</th> </tr> </thead> <tbody> <tr> <td>Yes</td> <td>Any</td> <td>Prepared, or heuristic action was taken</td> </tr> <tr> <td>No</td> <td>TR_ROLLBACK_RESOURCE_END</td> <td>Rolled back</td> </tr> <tr> <td>No</td> <td>TR_FORGET_RESOURCE_END</td> <td>Forgotten</td> </tr> <tr> <td>No</td> <td>Anything other than TR_ROLLBACK_RESOURCE_END or TR_FORGET_RESOURCE_END</td> <td>Committed</td> </tr> </tbody> </table>	XID Exists?	State	Inferred Status	Yes	Any	Prepared, or heuristic action was taken	No	TR_ROLLBACK_RESOURCE_END	Rolled back	No	TR_FORGET_RESOURCE_END	Forgotten	No	Anything other than TR_ROLLBACK_RESOURCE_END or TR_FORGET_RESOURCE_END	Committed
XID Exists?	State	Inferred Status														
Yes	Any	Prepared, or heuristic action was taken														
No	TR_ROLLBACK_RESOURCE_END	Rolled back														
No	TR_FORGET_RESOURCE_END	Forgotten														
No	Anything other than TR_ROLLBACK_RESOURCE_END or TR_FORGET_RESOURCE_END	Committed														

For information about manually resolving transactions, see [“Manually Resolving a Transaction”](#) on page 897.

Configuring XA Options in Integration Server

Using Integration Server, you can configure the following options for XA:

- Enabling or disabling XA transaction recovery.
- The location and initial size of the XA recovery store.
- Server parameters that determine:
 - The length of time that Integration Server waits between attempts to resolve a transaction.
 - The maximum time allowed to resolve a transaction.
 - Whether a transaction should continue if Integration Server cannot store the status of a transaction and its participating resources in the XA recovery store (for example, because the store is corrupted).
 - The maximum number of unique XA transactions in an XA recovery log file.

- Whether Integration Server writes transaction information to the XA recovery store.

The following sections provide more information about setting these options.

Enabling or Disabling XA Transaction Recovery

You can enable or disable XA transaction recovery. When you disable XA transaction recovery, Integration Server does not attempt to recover incomplete transactions automatically. Additionally, you cannot use Integration Server Administrator to recover incomplete transactions manually.

If you are willing to exchange reliability and recovery of XA transactions in return for possible improved processing performance, you might want to disable XA transaction recovery. By default, Integration Server enables XA transaction recovery.

➤ To enable or disable XA transaction recovery

1. Open the Integration Server Administrator if it is not already open
2. In Integration Server Administrator, go to **Settings > Resources** and click **XA Manual Recovery**.
3. Do one of the following:
 - If XA transaction recovery is currently enabled and you want to disable it, click **Disable XA Transaction Recovery**. Integration Server Administrator hides the **Unresolved XA Transaction** table.
 - If XA transaction recovery is currently disabled and you want to enable it, click **Enable XA Transaction Recovery**. Integration Server Administrator displays the **Unresolved XA Transaction** table.

Tip:

You can also use the `watt.server.jca.transaction.writeRecoveryRecord` server parameter to enable or disable XA transaction recovery. For more information about setting server parameters, see [“Server Configuration Parameters” on page 1017](#).

Configuring the XA Recovery Store

The XA recovery store is a persistent store that contains the XID and global state of a transaction. You can specify the location of the XA recovery store and the initial size of the file at start up.

➤ To configure the XA recovery store

1. Go to **Settings > Resources** and click **Store Settings**. In the **XA Recovery Store** section, Integration Server Administrator shows the current settings for the location of the XA recovery store and its initial size.

2. Click **Edit XA Recovery Store Settings**.
3. In the **Store Location** field, type the relative or absolute path to the directory in the file system in which to store the XA recovery store file. The default location is:

Integration Server_directory \instances\instance_name\XAStore

Important:

Make sure that you have write access to the specified directory and that the directory does not contain any characters considered illegal by your operating system.

4. In the **Initial Store Size (MB)** field, type the initial size for the XA recovery store file, in megabytes. The default is 10MB.

Note:

Make sure that there is enough free disk space on the Integration Server machine to accommodate the initial sizes of the default document store, the trigger document store, and the XA recovery store.

5. Click **Save Changes**.

When you next restart Integration Server, it will create a new XA recovery store file in the new location and start writing to it. Integration Server will also use the new initial size for the file.

Note:

In addition to specifying the initial size and location of the XA recovery store, you can use the `watt.server.transaction.xastore.maxTxnPerFile` to specify the maximum number of unique XA transactions in an XA recovery log file. For more information about this server parameter, see [“Server Configuration Parameters” on page 1017](#)

Configuring XA Server Parameters

Integration Server provides the following server parameters for XA transactions and XA transaction recovery.

watt.server.transaction.recovery.sleepInterval

If an error occurs while Integration Server is trying to resolve an uncompleted transaction, specifies the period of time Integration Server waits between resolution attempts.

watt.server.transaction.recovery.abandonTimeout

Specifies the maximum recovery time for resolving the transaction. Integration Server continues trying to resolve the transaction until the maximum recovery time expires.

watt.server.jca.transaction.rollbackOnWriteFailure

Specifies whether Integration Server should continue with a transaction or roll it back if Integration Server cannot store the status of a transaction and its participating resources in the XA recovery store (for example, because the store is corrupted). Setting the parameter to false involves some risk; if Integration Server ends abnormally, no statuses will have been saved to

the XA recovery store, and Integration Server will not be able to resolve the uncompleted transaction or give you the chance to resolve it manually.

watt.server.transaction.xastore.maxTxnPerFile

Specifies the maximum number of unique XA transactions in an XA recovery log file. When the XA recovery log file reaches the maximum number of transactions, Integration Server creates a new file. The default is 2000 transactions.

watt.server.transaction.xastore.performXALogging

Specifies whether Integration Server writes transaction information to the XA recovery store. Set to true to instruct Integration Server to log information about the state and progress of each XA transaction. Set to false to instruct Integration Server to skip logging XA transaction information. The default is true.

Important:

If you set this property to false, Integration Server does not log any information to the XA recovery store while processing a transaction, making transaction recovery impossible. If you want Integration Server to automatically resolve incomplete transactions or you want to manually resolve incomplete transactions, Integration Server must perform XA logging.

For more information about these and other server parameters, see [“Server Configuration Parameters” on page 1017](#).

Manually Resolving a Transaction

If Integration Server cannot resolve a transaction, you can try to resolve it manually. To successfully resolve a transaction manually, you must be familiar with the participating resources and must act in a way that leaves the transaction in a consistent state. For example, if only one of the participating resources committed the transaction, you can try to get the other participating resources to commit as well.

When you manually resolve a transaction, resolution is not itself a transaction; that is, each participating resource and the action you perform on it does not participate in a new 2PC transaction. You must therefore make sure your actions result in a consistent state for the participating resources.

➤ **To manually resolve an XA transaction**

1. Open the Integration Server Administrator if it is not already open
2. In Integration Server Administrator, go to **Settings > Resources** and click **XA Manual Recovery**. Integration Server Administrator displays all of the unresolved XA transactions. For a description of the information displayed for each unresolved transaction, see [“About Unresolved XA Transactions” on page 892](#).

Note:

Refresh the page at intervals to make sure all uncompleted transactions are listed. Suppose Integration Server tries to resolve an uncompleted transaction but cannot; the transaction

will not be listed while Integration Server is trying to resolve it, but if you refresh the page later, the transaction will appear on the list.

- In the **XID** column, click the XID for the transaction that you want to resolve. The Integration Server Administrator displays detailed information about the resources involved in the transaction. For a description of the information displayed for each participating resource, see [“Details for an Unresolved XA Transaction” on page 893](#).
- If you want to delete the transaction, click the **Delete Transaction** link. Deleting the transaction removes the transaction from the XA recovery store.

You might want to simply delete a transaction if you do not want to resolve a transaction using Integration Server Administrator (for example, because you want to resolve the transaction by working with the resources directly).

- If you want to resolve the transaction using Integration Server Administrator, select one of the following in the **Desired Action** column.

If you want to	Select
You want to commit the transaction on the resource	Commit
You want to roll back the transaction on the resource	Roll back
The resource administrator heuristically committed or rolled back the transaction, so you want to erase the XID from the resource	Forget
The resource administrator has already taken the correct action on the resource so you need take none, or the resource is down for an extended period	Do nothing

Note:

The **Desired Action** column lists the possible actions for each resource, based on the combination of the values for **State** and **XID** for the resource, and selects the most logical action by default.

- Click **Perform Action**. Integration Server Administrator returns to the **XA Manual Recovery** screen and removes the transaction from the list of unresolved transactions.

Integration Server might receive and display an error from a resource. Errors can occur for these reasons:

- The resource was not connected to Integration Server, probably because the resource was down.

- The resource has no knowledge of the transaction, possibly because it lost the 2PC transaction information.
- The resource threw an exception.
- The transaction involved a webMethods adapter, and Integration Server cannot locate the resource because someone deleted or changed the adapter connection node that pointed to the resource from Software AG Designer.

You might have to force the transaction to a consistent state using the tools available on the resource itself.

49 Content Handlers in Integration Server

■ How Integration Server Uses Content Handlers	902
■ How Integration Server Chooses a Content Handler for an HTTP Request	902
■ How Integration Server Chooses a Content Handler for an FTP Request	902
■ Content Handlers for HTTP and FTP Requests	902
■ About Content Handlers for HTTP Responses	908
■ About Content Handlers for FTP Responses	909

How Integration Server Uses Content Handlers

When a client submits an HTTP or FTP request, Integration Server uses a content handler to parse the contents of the request and pass them to the target service. Integration Server also uses a content handler when building the HTTP or FTP response.

How Integration Server Chooses a Content Handler for an HTTP Request

To determine which content handler to use for an incoming HTTP request, Integration Server looks at the Content-Type header field in the request header, and then uses the content handler registered for it. For example, if the request header specifies Content-Type=text/html, Integration Server looks for a content handler that has been registered as the text/html content handler.

How Integration Server Chooses a Content Handler for an FTP Request

To determine which content handler to use for an incoming FTP request, the FTP listening port on Integration Server chooses the content handler based on the file extension of the request. The *Integration Server_directory/instances/instance_name/lib/mime.types* file contains the mappings of file extension to content type.

You can edit the mappings in the lib/mime.types file from Integration Server Administrator by selecting **Settings > Resources > Mime types**. For more information about editing the mappings of the lib/mime.types file, see *webMethods Service Development Help*.

Content Handlers for HTTP and FTP Requests

Integration Server selects a content handler based on the content type of the request and uses the content handler to parse the content of the request.

The following table identifies which content handler Integration Server uses for incoming HTTP and FTP requests based on the registered content type.

Integration Server uses the content handler	When the content type is	To parse
ContentHandler_CGI	text/html and watt.server.default ContentHandler is set to false	CGI content and unformatted text. This content handler does the following: <ul style="list-style-type: none">■ Decodes an incoming input stream and converts the content to key/value pairs. If the key already exists, the value is added

Integration Server uses the content handler	When the content type is	To parse
ContentHandler_Default	text/html and watt.server.default ContentHandler is set to true	<p data-bbox="950 294 1356 357">to a list in the pipeline with the key <i>key+list</i>.</p> <ul data-bbox="901 388 1356 451" style="list-style-type: none"> <li data-bbox="901 388 1356 451">■ Writes the key/value pair to an HTML table. <p data-bbox="901 483 1390 546">Name/value pairs obtained from web forms.</p> <p data-bbox="901 577 1390 640">Input stream from an HTTP request or a document from an FTP request.</p> <p data-bbox="901 672 1291 735">This content handler does the following:</p> <ul data-bbox="901 766 1390 1291" style="list-style-type: none"> <li data-bbox="901 766 1390 966">■ For HTTP requests, it passes the contents of the input stream from the HTTP request to the service specified in the submit method in an input stream named <i>contentStream</i>. <li data-bbox="901 997 1390 1197">■ For FTP requests, it passes the contents of the document from the FTP request to the service specified in the submit method in an input stream named <i>contentStream</i>. <li data-bbox="901 1228 1390 1291">■ For HTTP and FTP, it writes a key/value pair to an HTML table.
ContentHandler_FlatFile	application/x-wmflatfile	<p data-bbox="901 1323 1112 1354">Flat file content.</p> <p data-bbox="901 1386 1291 1449">This content handler does the following:</p> <ul data-bbox="901 1480 1390 1711" style="list-style-type: none"> <li data-bbox="901 1480 1390 1617">■ Passes the contents of the flat file to the service specified in the submit method in an input stream named <i>ffdata</i>. <li data-bbox="901 1648 1390 1711">■ Returns data in an input stream or a ByteArray called <i>ffreturn</i>. <p data-bbox="901 1743 1390 1837">For information about using this content-type header, see <i>Flat File Schema Developer's Guide</i>.</p>

Integration Server uses the content handler	When the content type is	To parse
ContentHandler_IDAT	application/x-wmidatabin	<p data-bbox="808 289 1289 363">Input stream from an HTTP request or a document from an FTP request.</p> <p data-bbox="808 384 1289 457">This content handler does the following:</p> <ul data-bbox="808 478 1289 646" style="list-style-type: none"> <li data-bbox="808 478 1289 552">■ Converts the input stream or document to an iData object. <li data-bbox="808 573 1289 646">■ Converts the data to an XML representation of IData.
ContentHandler_JSON	application/json	<p data-bbox="808 667 1289 701">JSON content.</p> <div data-bbox="813 716 1273 888" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="813 730 1273 867">Note:Integration Server supports the application/json content type only with the admin, invoke, rest, and restv2 directives.</p> </div> <p data-bbox="808 905 1289 1041">The behavior of this content handler is based on the setting for the <code>watt.server.http.jsonFormat</code> parameter.</p> <ul data-bbox="808 1062 1289 1745" style="list-style-type: none"> <li data-bbox="808 1062 1289 1314">■ When set to <code>parsed</code>, parses JSON content into pipeline variables automatically. You can then use JSON to compose pipelines and execute existing services without calling <code>pub.json:jsonStreamToDocument</code>. <li data-bbox="808 1335 1289 1608">■ When set to <code>stream</code>, adds the request body to the pipeline as <code>jsonStream</code>. The <code>pub.json:jsonStreamToDocument</code> service converts the <code>jsonStream</code> into a document (IData object) so you can use the elements from <code>jsonStream</code> in a flow service. <li data-bbox="808 1629 1289 1745">■ When set to <code>bytes</code>, passes the incoming stream of JSON content as a byte array named <code>jsonBytes</code>. <div data-bbox="813 1759 1273 1896" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="813 1774 1273 1875">Note:For detailed information about the <code>watt.server.http.jsonFormat</code></p> </div>

Integration Server uses the content handler	When the content type is	To parse
ContentHandler_Multipart	multipart/related multipart/mixed multipart/form-data	<p data-bbox="901 283 1372 556">parameter, see “Server Configuration Parameters” on page 1017. For more information about the <code>pub.json:jsonStreamToDocument</code> service, see <i>webMethods Integration Server Built-In Services Reference</i>.</p> <p data-bbox="901 567 1372 745">Objects that are part of related body parts. For example, an HTML web page with its related <code>/graphics/administrator_guide_graphics/graphics</code> files.</p> <p data-bbox="901 766 1372 840">This content handler does the following:</p> <ul data-bbox="901 861 1372 1060" style="list-style-type: none"> ■ Passes the input stream to the pipeline with the key <code>contentStream</code>. ■ Writes a key/value pair to an HTML table.
ContentHandler_RPC	application/x-wmrpc	<p data-bbox="901 1081 1372 1155">Input stream from an HTTP request or a document from an FTP request.</p> <p data-bbox="901 1176 1372 1249">This content handler does the following:</p> <ul data-bbox="901 1270 1372 1438" style="list-style-type: none"> ■ Converts the data from an input stream to a key/ value pair. ■ Writes the values as RPC encoded to the data stream.
ContentHandler_RPC2	application/x-wmrpc2	<p data-bbox="901 1459 1372 1533">Input stream from an HTTP request or a document from an FTP request.</p> <p data-bbox="901 1554 1372 1627">This content handler does the following:</p> <ul data-bbox="901 1648 1372 1808" style="list-style-type: none"> ■ Creates an <code>iData</code> object from the input stream or document. ■ Writes the values as RPC encoded to the data stream.

Integration Server uses the content handler	When the content type is	To parse
ContentHandler_SOAP	text/xml	<p>Input stream from an HTTP request or a document from an FTP request.</p> <p>This content handler does the following:</p> <ul style="list-style-type: none"> ■ Adds a SOAP request message to the pipeline with the key <i>soapRequestData</i>. ■ Adds a SOAP response to the pipeline with the key <i>soapResponseData</i>.
	application/soap	SOAP messages.
	application/soap+xml	SOAP 1.2 messages serialized as XML.
ContentHandler_XML	text/xml	<p>XML content.</p> <p>Note:Integration Server supports the text/xml content type only with the <code>invoke</code>, <code>rest</code>, and <code>restv2</code> directives.</p> <p>The behavior of this content handler is based on the setting of the <code>watt.server.coder.responseAsXML</code> parameter. This behavior is applicable only for REST V2 requests.</p> <ul style="list-style-type: none"> ■ When set to <code>true</code>, Integration Server receives the XML response for any REST API request in proper XML format. ■ When set to <code>false</code>, Integration Server receives the XML response for any REST API request in converted XML format containing IData objects.
	application/xml	<p>XML content.</p> <p>Note:Integration Server supports the application/xml content type</p>

Integration Server uses the content handler	When the content type is	To parse
		only with the <code>invoke</code> , <code>rest</code> , and <code>restv2</code> directives.
		The behavior of this content handler is based on the setting of the <code>watt.server.coder.responseAsXML</code> parameter. This behavior is applicable only for REST requests.
		<ul style="list-style-type: none"> ■ When set to <code>true</code>, Integration Server receives the XML response for any REST API request in proper XML format. ■ When set to <code>false</code>, Integration Server receives the XML response for any REST API request in converted XML format containing <code>IData</code> objects.

Notes:

- Other webMethods products and customer applications can register different content handlers for any of these Content-Type headers, which might change the behavior of Integration Server when Integration Server receives messages of that type. For example, Trading Networks registers its own XML content handler for `text/xml`. When this happens, Integration Server will use the Trading Networks content handler to process incoming XML messages.
- If the Content-Type header field specifies a content type for which no content handler has been registered, Integration Server uses the default content handler (`ContentHandler_Default`), which treats the content as `text/html`.
- If an FTP or FTPS port receives a file that does not have an extension, Integration Server uses the default content handler (`ContentHandler_Default`), which treats the content as `text/html`.
- Integration Server provides the following Java API that enables you to register your own services along with the associated coders for multipart content handler:
`com.wm.app.b2b.server.ServerAPI.registerCoderForMultipart(String serviceName, Class coder)`. You can unregister the coder using the following Java API:
`com.wm.app.b2b.server.ServerAPI.removeCoderForMultipart(String serviceName)`. If you do not register a coder or when you unregister an existing coder for a particular service, then Integration Server uses the default content handler (`ContentHandler_Default`) for the service.
- A web service request that specifies a content type of `application/xml` succeeds in versions of Integration Server prior to version 9.12. However, the web service request fails in Integration Server version 9.12 and later. This is because prior to Integration Server 9.12, Integration Server did not provide out of the box support to directly handle the content type `application/xml`.

If a content handler was not specified for application/xml, Integration Server used the default content handler. The default handling is the same as the handling of text/html, resulting in XML being placed in the pipeline as an inputStream. Beginning with version 9.12, Integration Server added a content handler named ContentHandler_XML to support the content type application/xml, resulting in the content being placed in the pipeline as an XML node. This behavior change causes the web service request to fail in version 9.12 because the web service expects an inputStream, not an XML node. To address this issue and avoid having to change the service to accept an XML node instead of an inputStream, you can use the `watt.server.content.type.mappings` server configuration parameter to map the application/xml content type to the text/xml content type. Integration Server will use the text/xml content handler when receiving a request that specifies a content type of application/xml. Specify the following:

```
watt.server.content.type.mappings=application/xml text/xml
```

About Content Handlers for HTTP Responses

Integration Server does the following to select a content handler for an HTTP response:

1. Looks at the Content-Type header field in the response header, and then uses the contentHandler registered for that content type. This field has a value only if the application assigns one using the `pub.flow:setResponseHeader` service.

If the Content-Type header field in the response header specifies an unsupported content type, Integration Server uses text/html.

2. If content type is not specified in the response header, Integration Server looks at the Accept header field in the request header to determine content type, and then uses the content handler registered for that content type.
3. If there is no Accept header field in the request header, or if there is no content handler for the content-type specified by the Accept header field, Integration Server uses the content handler that was used for the request.

Note:

If a response has already been created by the `pub.flow:setResponse2` service or `pub.flow:HTTPResponse` document type, Integration Server does not use a content handler for the response.

Accept Header Field

The Accept header field specifies which content type or types the client will accept in the response. The Accept header field can specify multiple acceptable content types, either by listing them (application/json, text/html, text/xml, and so on), or by using wildcards (text/*).

By default, if the Accept header specifies a content type using a wildcard, Integration Server will select the first content handler registered for that major type (text, application, multipart).

The following table identifies the content handler that Integration Server selects based on the value of the Accept header field.

This Accept Header field	Results in this content handler
text/*	XML
application/*	CGI
multipart/*	Multipart

To control which content handler Integration Server will use when a wildcard is specified in the Accept header, you must use the `watt.server.content.type.mappings` server configuration parameter. The syntax for this parameter is:

```
watt.server.content.type.mappings=<wildcard1> <content-type1>,<wildcard2>
<content-type2>,<wildcardN> <content-typeN>
```

For example, to associate `text/*` with `text/xml` and `multipart/*` with `multipart/related`, specify the parameter as follows:

```
watt.server.content.type.mappings=text/* text/xml,multipart/* multipart/related
```

The Accept header might contain multiple content types with parameters indicating an order of preference. Integration Server will attempt to respond with the "most preferred" content type as defined by [RFC 2616, section 14](#).

Integration Server supports the Accept header field in HTTP request headers if the `watt.server.http.useAcceptHeader` configuration parameter is set to `true`. The default setting for this parameter is `true`.

About Content Handlers for FTP Responses

When creating a response, Integration Server uses the content handler registered for the content type in the `lib/mime.types` file.

The `Integration Server_directory/instances/instance_name/lib/mime.types` file contains the mappings of file extension to content type. You can edit the mappings with Integration Server Administrator by selecting **Settings > Resources > Mime types**. For more information about editing the mappings of the `lib/mime.types` file, see *webMethods Service Development Help*.

Note:

If a response has already been created by the `pub.flow:setResponse2` service, Integration Server does not use a content handler for the response.

50 Class Loading in Integration Server

■	How Class Loading Works in Integration Server	912
■	Adding Classes to the Server Classpath	917
■	Where to Put Your Classes and Jar Files	922
■	Accelerating Class Loading	923

How Class Loading Works in Integration Server

Integration Server runs in the Java Virtual Machine (JVM). When Integration Server, or any Java application, needs access to a class, the JVM locates the executable bytecode of the class and brings it into the JVM. This process is referred to as *loading the class* and is handled by class loaders. The class loaders in Integration Server and the JVM are designed to load classes from different locations.

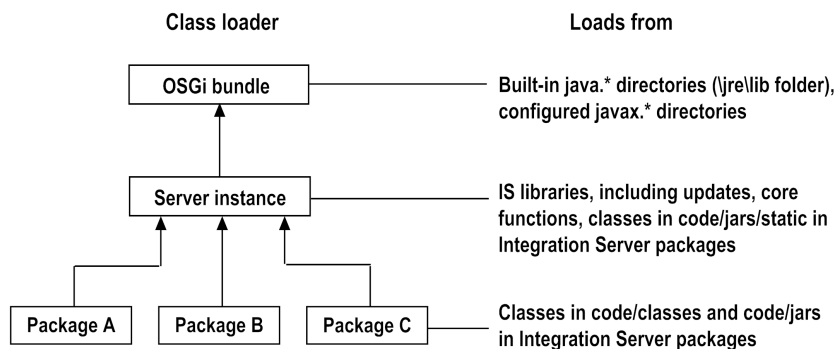
During server initialization, at the point when Integration Server loads packages, Integration Server also loads all classes that contain Java services. A Java service contains statements that reference other Java classes. To execute these statements, the referenced classes must be loaded into the JVM.

If you are using your own classes developed by your company or classes from a third-party library, you can control the visibility and loading order of these classes. Class visibility refers to whether the classes and jar files will be or need to be visible to all the classes in Integration Server or to a particular package only.

Class Loaders

Integration Server runs on top of the OSGi platform supported by Eclipse Equinox. The parent class loader of Integration Server is the OSGi bundle class loader.

The following diagram shows the different class loaders and their relationship in the class loader chain:



Note: Microservices Runtime does not use OSGi. The parent classloader for Microservices Runtime is the Java System Class Loader.

OSGi Bundle Class Loader

The OSGi bundle classloader is the parent class loader for Integration Server. The OSGi bundle classloader is provided by the OSGi framework (Eclipse Equinox) and shipped with Integration Server.

Note:

The parent classloader for Microservices Runtime is the Java System Class Loader.

By default, the OSGi bundle classloader cannot access any jars specified in the CLASSPATH environment variable or in some Java specific parameters (for example, the `-javaagent` parameter). If you want to access jars specified in either the CLASSPATH environment variable or a Java specific parameter, you must add the following entry in the *Software AG_directory* `\profiles\IS_instance_name\config.ini` file as follows:

```
osgi.parentClassLoader=app
```

Note:

After you add the `osgi.parentClassLoader` property to the `config.ini` file, you must restart Integration Server.

Integration Server Class Loaders

Integration Server includes a server class loader and package class loaders. These class loaders load classes required for Integration Server to function.

- Integration Server Server Class Loader loads the classes that comprise the core of Integration Server. This loader loads from the **Server Classpath**, which you can see on the About page, and which is described below.
- Integration Server Package Class Loaders load classes from Integration Server packages. These packages include those created by users through Designer and predefined packages that are provided as part of Integration Server. Each package has its own class loader.

Classpaths

The Integration Server classpath is the list of files and directories that Integration Server class loaders search when looking for classes. In Integration Server Administrator, the About screen shows the files and directories that make up the classpaths used by the Integration Server's class loaders. The following image shows a portion of that display.

Java Classpath	C:\SoftwareAG\common\lib\tw-3.5.25\wrapper.jar C:\SoftwareAG\common\runtime\bundles\platform\eclipse\plugins\com.softwareag.platform.bootstrap_9.10.0.0000-0436.jar C:\SoftwareAG\common\runtime\bundles\platform\eclipse\plugins\org.eclipse.equinox.launcher_1.3.100.v20150511-1540.jar C:\SoftwareAG\Integration Server\instances\...\common\lib\wm-converters.jar
	C:\SoftwareAG\Integration Server\instances\...\common\lib\wm-converters.jar C:\SoftwareAG\Integration Server\instances\default\lib\classes\ C:\SoftwareAG\Integration Server\lib\wm-isserver.jar C:\SoftwareAG\common\lib\wm-isclient.jar C:\SoftwareAG\common\lib\castor-all.jar C:\SoftwareAG\common\lib\ehcache-ee.jar C:\SoftwareAG\common\lib\glue.jar C:\SoftwareAG\common\lib\saglic.jar C:\SoftwareAG\common\lib\terracotta-toolkit-runtime-ee.jar C:\SoftwareAG\common\lib\wm-acdi-common.jar C:\SoftwareAG\common\lib\wm-bpm-processmodel.jar C:\SoftwareAG\common\lib\wm-caf-client.jar C:\SoftwareAG\common\lib\wm-caf-common.jar C:\SoftwareAG\common\lib\wm-caf-event-common.jar C:\SoftwareAG\common\lib\wm-caf-jsf.jar C:\SoftwareAG\common\lib\wm-caf-rules.jar C:\SoftwareAG\common\lib\wm-caf-server.jar C:\SoftwareAG\common\lib\wm-calendar-components.jar C:\SoftwareAG\common\lib\wm-calendar.jar C:\SoftwareAG\common\lib\wm-cdc-core.jar C:\SoftwareAG\common\lib\wm-directory-components.jar C:\SoftwareAG\common\lib\wm-directory.jar C:\SoftwareAG\common\lib\wm-eform-common.jar C:\SoftwareAG\common\lib\wm-eform.jar C:\SoftwareAG\common\lib\wm-g11nutils.jar C:\SoftwareAG\common\lib\wm-lwq.jar C:\SoftwareAG\common\lib\wm-metadata-core-centrasite.jar C:\SoftwareAG\common\lib\wm-metadata-core.jar C:\SoftwareAG\common\lib\wm-mws-library.jar C:\SoftwareAG\common\lib\wm-scg-audit.jar C:\SoftwareAG\common\lib\wm-scg-cluster.jar C:\SoftwareAG\common\lib\wm-scg-core.jar C:\SoftwareAG\common\lib\wm-scg-jdbcpool.jar

How the Integration Server Classpath Is Set

When you start Integration Server, a script file named `startup.bat` (on Windows) or `startup.sh` (on UNIX) runs. Among other things, the `startup.bat/sh` file builds an Integration Server classpath variable which consists of many directories and variables.

The following table describes the directories and variables that Integration Server uses to construct the Integration Server classpath.

Directory or variable	Description
The <code>wrapper.java.additional.202=</code> <code>-Dwatt.server.prepend.classes</code> property of the <i>Software AG_directory</i> /profiles/ IS_instance_name/ configuration/ custom_wrapper.conf	Variable defined in <code>custom_wrapper.conf</code> that defines directories and files to prepend to the beginning of the classpath.
The <code>wrapper.java.additional.203=</code> <code>-Dwatt.server.append.classes</code> property of the <i>Software AG_directory</i> /profiles/ IS_instance_name/ configuration/ custom_wrapper.conf	Variable defined in <code>custom_wrapper.conf</code> that defines directories and files to append to the end of the classpath.

Directory or variable	Description
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \\ \textit{Integration Server_directory} \\ \\ \backslash \end{array}$ \instances\instance_name\bin\ini.cnf	The application.classpath property from <i>Integration Server_directory</i> \instances\instance_name\bin\ini.cnf. This typically specifies classes in the following directories, in this order: $\begin{array}{l} \textit{Integration Server_directory} \backslash \textit{lib} \\ \\ \textit{Software AG_directory} \backslash \textit{common} \backslash \textit{lib} \end{array}$
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \\ \textit{Integration Server_directory} \\ \backslash \\ \textit{instances} \backslash \textit{instance_name} \backslash \textit{lib} \backslash \textit{jars} \backslash \textit{custom} \end{array}$	All custom or third party .jar and .zip files in this directory. Note: Use this directory to store your custom or third party .jar and .zip files that you want to make available to a server specific instance.
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \\ \textit{Integration Server_directory} \\ \backslash \\ \textit{instances} \backslash \textit{instance_name} \backslash \textit{packages} \backslash \textit{package_name} \backslash \textit{code} \backslash \textit{jars} \backslash \textit{static} \end{array}$	All .jar and .zip files from this directory in each package. The files are added to the classpath in the order returned by the File.list() method. Note: Jars contained in this directory will be loaded even if the associated package is disabled.
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \\ \textit{Integration Server_directory} \\ \\ \backslash \end{array}$ \instances\instance_name\packages\package_name\code\classes	If this directory exists, it is added to the classpath.
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \\ \textit{Integration Server_directory} \\ \\ \backslash \end{array}$ \instances\instance_name\packages\package_name\code\classes.zip	If this file exists, it is added to the classpath.
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \\ \textit{Integration Server_directory} \\ \backslash \end{array}$ \instances\instance_name\updates	Valid, and non-voided updates and fixes.
$\begin{array}{l} \textit{Software AG_directory} \\ \backslash \end{array}$	All .jar and .zip files in this directory.

Directory or variable	Description
<i>Integration Server_directory</i> lib\jars	
<i>Software AG_directory</i> \ <i>Integration Server_directory</i> lib\jars\custom	Common custom and third party .jar and .zip files in this directory. Note: Use this directory to store your custom and third party .jar and .zip files that you want to make available to all server instances.
<i>Software AG_directory</i> \ <i>Integration Server_directory</i> updates	Common valid, and non-voided updates and fixes. Note: The updates and fixes in this directory are available to all server instances.

Note:

In some cases, the Integration Server Administrator About page will display the names of files that do not exist. If filenames specified in the `wrapper.java.additional.202=-Dwatt.server.prepend.classes` and `wrapper.java.additional.203=-Dwatt.server.append.classes` properties for the Integration Server classpath do not exist, they will still be displayed on the Integration Server About page. If the `ini.cnf` file refers to jar files that do not exist, they will also be displayed on the About page and, if the `watt.debug.warnOnClasspathError` server configuration parameter is set to true, the following error will be written to the standard out for each non-existent jar file: `Classpath entry in ini.cnf not found: <jar_file_name>`. Standard out is directed to *Software AG_directory*/profiles/IS_instance_name/logs/wrapper.log.

Additional Information about How the Microservices Runtime Classpath Is Set

The Microservices Runtime classpath is set in much the same way as the Integration Server classpath. However, there are two significant differences:

- Microservices Runtime does not use instances. References to *Software AG_directory* \
Integration Server_directory \`instances` \`instance_name` for Integration Server in the table in [“How the Integration Server Classpath Is Set”](#) on page 914 correspond to just *Integration Server_directory* for Microservices Runtime.
- Microservices Runtime does not use OSGi and, therefore, does not use information from the `custom_wrapper.conf` file to build the classpath. Instead, Microservices Runtime uses information from *Integration Server_directory* /bin/setenv.bat(sh), specifically from the `PREPENDCLASSES` and `APPENDCLASSES` properties.

Note:

In some cases, the Microservices Runtime About page will display the names of files that do not exist. Specifically, if filenames specified in the `PREPENDCLASSES` and `APPENDCLASSES` variables for the Microservices Runtime classpath do not exist, they will still be displayed on the Microservices Runtime About page.

Adding Classes to the Server Classpath

If you are using your own classes developed by your company or classes from a third-party library, you can control the visibility and loading order of these classes. Class visibility refers to whether the classes and jar files will be or need to be visible to all the classes in Integration Server or to a particular package only.

Classes and jar files that are added to the server classpath are visible to all classes in Integration Server including all Java services. In a sense, the classes and jar file are globally visible.

Note:

The visibility of a class may also be restricted to members of the same Java package by omitting the "public" modifier from the class declaration. This is not directly related to class loading and is described in the Java documentation.

There are three ways to add classes and libraries (jar and zip files) to the classpath.

- **Option 1.** Copy jar or zip files to a package's `code\jars\static` directory. Integration Server adds the files to the classpath when the package loads.

Note that if you change the contents of a jar file in a package's `code\jars\static` directory, you must restart Integration Server for the change to take effect. Simply reloading the package is not sufficient.

- **Option 2.** Copy jar or zip files to the `Software AG_directory \ Integration Server_directory \ instances\instanceName\lib\jars\custom` directory. Integration Server adds the files to the classpath when Integration Server starts.

If you want to add your jar or zip files to the classpath for all Integration Server instances in an installation, copy the files to the `Software AG_directory \ Integration Server_directory \ lib\jars\custom` directory.

Note:

For Option 2 for Microservices Runtime, add your jar or zip files to `Software AG_directory \ Integration Server_directory \ lib\jars\custom`

- **Option 3.** Copy files to another location on the file system. Add the path to the jar or class files directory to the `watt.server.prepend.classes` or `watt.server.append.classes` properties set in the following file:

`Software AG_directory \ profiles\IS_instanceName\configuration\custom_wrapper.conf:`

If the `custom_wrapper.conf` file contains overrides to the classpath information, the server uses those settings during startup.

The following table identifies the variables you can use to add directories to the beginning and the end of the Integration Server classpath:

To add directories to the	Edit this variable
Beginning of the classpath	<code>wrapper.java.additional.202=</code> <code>-Dwatt.server.prepend.classes=</code>
End of the classpath	<code>wrapper.java.additional.203=</code> <code>-Dwatt.server.append.classes=</code>

For Microservices Runtime, add the path to the jar or class files to: *Integration Server_directory* /bin/setenv.bat(sh)

The following table identifies the Java system properties that you can set in setenv.bat(sh) to add directories the beginning or end of the classpath for Microservices Runtime.

To add directories to the	Edit this Java system property
Beginning of the classpath	PREPENDCLASSES in setenv.bat/sh.
End of the classpath	APPENDCLASSES in setenv.bat/sh.

For more detailed information about where and how these variables are added to the classpaths, see [“Classpaths” on page 913](#). For more information about the custom_wrapper.conf and the properties you can set, see *Software AG Infrastructure Administrator’s Guide*.

Keep the following information in mind when choosing how to add classes and libraries to the classpath.

- Options 1 and 2 described above will work only for jar and zip files. Option 3 will work for jar, zip, class, and other files.
- Some third-party libraries look in the classpath for configuration or other types of files. Use option 3 to add these files to the classpath.
- When option 1 is used, all jar and zip files in a package's code\jars\static directory are automatically included if the package is deployed to another Integration Server. With options 2 and 3, an additional step is required to deploy the jar, zip or class files.
- Where the classes appear on the server classpath affect the loading order. For more information about loading order, see [“How Classes Are Found” on page 920](#)

Adding Classes to a Package

If a class will be used only by services in a single Integration Server package, you can limit the visibility of the class to that package by placing the class in the package's code/classes directory. If the class is distributed in a jar or zip file, place it in the package's code/jars directory. Files in a package's code/classes or code/jars directories are not added to the Integration Server’s classpath.

Note:

If you change the contents of a jar or class file in a package's code/jars or code/classes directory, reload the package to make the change take effect. You do not need to restart Integration Server for the changes to take effect.

If you have a package that needs access to a class in another package's code/classes or code/jars directory, you can provide access by creating a dependency relationship between the two packages. For example, consider the following scenario:

- PackageA has a class in its code/classes directory.
- PackageB needs to use the class in PackageA's code/classes directory.

You can use Designer to establish a dependency on PackageA from PackageB.

Alternatively, you can modify PackageB's manifest.v3 to require PackageA.

```
<record name="requires" javaclass="com.wm.util.Values">
<value name="PackageA">1.0</value>
</record>
```

After editing the manifest, if Integration Server is running, reload PackageB.

PackageB is now able to access classes in PackageA's code/classes and code/jars directories. Because PackageB requires PackageA, PackageB cannot be enabled unless PackageA is enabled. PackageA cannot be disabled unless PackageB is disabled. Reloading PackageA will cause PackageB to be reloaded.

Class Searching Order

When a Java service refers to a class, Integration Server will, by default, look for the referenced class in this order:

1. On the server classpath.
2. Locally, within the package.
3. In all required packages.

When a class is found, Integration Server immediately loads and uses the class.

You can change the class loading order by modifying a package's manifest.v3 file to use the package class loader:

```
<value name='classloader'>package</value>
```

When a package's classloader property is "package", the class loading order is as follows:

1. Locally, within the package.
2. In all required packages.
3. On the server classpath.

The default value for the `classloader` property is "system". That is what Integration Server uses when no value is set for the `classloader` property:

```
<value name='classloader' />
```

For detailed information about configuring a package to use the package class loader, see [“Using a Package Class Loader” on page 697](#).

How Classes Are Found

When you are choosing where to put your classes on the server classpath, as described in [“Adding Classes to the Server Classpath” on page 917](#), consider the effect of the possible locations on the loading order. Here is the order in which Integration Server searches the various locations relative to one another:

1. `watt.server.prepend.classes`
2. `Software AG_directory \ Integration Server_directory \ instances \ instanceName \ bin \ ini.cnf`
Specifically, the `application.classpath` property in the `ini.cnf` file. The `application.classpath` property typically specifies classes in the following directories, in this order:
 - `Integration Server_directory \ lib`
 - `Software AG_directory \ common \ lib`
3. `Software AG_directory \ Integration Server_directory \ instances \ instanceName \ lib \ jars`
4. `Software AG_directory \ Integration Server_directory \ instances \ instanceName \ lib \ jars \ custom`
5. `Software AG_directory \ Integration Server_directory \ lib \ jars`
6. `Software AG_directory \ Integration Server_directory \ lib \ jars \ custom`
7. `Software AG_directory \ Integration Server_directory \ instances \ instanceName \ packages \ packageName \ code \ jars \ static`
8. `watt.server.append.classes`

Note:

For information about the directories Microservices Runtime searches for class and jar files, see [“How Classes Are Found in Microservices Runtime ” on page 921](#).

If you have two or more copies of a class, whether it is a single class file or is a class in a jar file, and multiple locations for the class appear in the server classpath, Integration Server always loads the instance of the class that appears first in the server classpath. Integration Server never uses other instances of the same class that appear later in the server classpath. A scenario where this is significant is when different versions of a third-party jar file are on the server classpath. Integration Server always uses the version that comes first in the server classpath.

If more than one package has jar files in its `code \ jars \ static` directory, the jar files will be added to the server classpath in alphabetical order, by package name. For example, jar files in `PackageA \ code \ jars \ static` will be loaded before jar files in `PackageB \ code \ jars \ static`.

You can view the Integration Server classpath on the About page in Integration Server Administrator.

How Classes Are Found in Microservices Runtime

The locations that Microservices Runtime searches relative to one another is different than Integration Server because Microservices Runtime does not use OSGi and does not have a “\instances*instanceName*” directory.

Below is the order in which Microservices Runtime searches the various locations relative to one another:

1. PREPENDCLASSES in *Integration Server_directory* \bin\server.bat(sh)
2. *Software AG_directory* \ *Integration Server_directory* \bin\ini.cnf

Specifically, the application.classpath property in the ini.cnf file. The application.classpath property typically specifies classes in the following directories, in this order:

- *Integration Server_directory* \lib
 - *Software AG_directory* \common\lib
3. *Software AG_directory* \ *Integration Server_directory* \lib\jars
 4. *Software AG_directory* \ *Integration Server_directory* \lib\jars\custom
 5. *Software AG_directory* \ *Integration Server_directory* \packages*packageName*\code\jars\static
 6. APPENDCLASSES in *Integration Server_directory* \bin\server.bat(sh)

Note:Microservices Runtime handles two or more copies of a class or multiple packages with jar files in its code\jars\static directory the same way in which Integration Server does. For more information, see [“How Classes Are Found” on page 920](#).

Searching Order within a Package

When Integration Server searches within a package for a class, it uses this order:

1. *packageName*\resources
2. *packageName*\lib
3. *packageName*\code\jars
4. *packageName*\code\classes

Note:

When a package requires another package, Integration Server will first look for a referenced class in the above order within the package that refers to the class. Then Integration Server will look in the above order in the required packages.

Where to Put Your Classes and Jar Files

If you are using your own classes developed by your company or classes from a third-party library, you can control the visibility and loading order of these classes. Class visibility refers to whether the classes and jar files will be or need to be visible to all the classes in Integration Server or to a particular package only.

Use the following guidelines to help you decide where to put your class files and jar files.

- If a class or jar file is needed only by services in one Integration Server package, put it in the package's code/classes or code/jars directory.
- If a class or jar file is needed by a second Integration Server package, keep the class or jar in the package that it already resides in and establish a package dependency on the first package. You can accomplish this by using Designer to establish a dependency on the first package from the second package or by modifying the second package's manifest.v3 file to require the first package.
- If additional Integration Server packages need the class or jar file, and the package dependencies are becoming difficult to manage, do one of the following:
 - If it is a jar or zip file, move it to the package's code/jars/static directory. This will add the jar or zip to the server's classpath and make it available to all packages.
 - If it is a class or a set of classes and you are using Integration Server, add the full path to the parent directory in one of these lines in the following file: *Software AG_directory \profiles\IS_instanceName\configuration\custom_wrapper.conf*

```
wrapper.java.additional.202=-Dwatt.server.prepend.classes=  
wrapper.java.additional.203=-Dwatt.server.append.classes=
```
- If it is a class or a set of classes and you are using Microservices Runtime, add the full path to the parent directory in one of the Java system properties in the following file: *Software AG_directory \Integration Server_directory \bin\setenv.bat(sh)*

```
APPENDCLASSES  
PREPENDCLASSES
```
- If your package needs a different version of a class or jar file than one that resides in another package, copy the version you need to your package's code/classes or code/jars directory.
- If your package needs a different version of a class or jar file than one that is on the server classpath, do the following:
 1. Copy the version you need to your package's code/classes or code/jars directory.
 2. Change your package's classloader property to "package". For more information about
- When a Java security provider is registered with the JVM, the provider is available to all Integration Server packages. Java security is implemented in such a way that the JVM retains the implementation of the provider along with its class loader and makes this implementation

available to every process running within the JVM. This affects other Integration Server packages as well as any other installed products that run in the JVM. For this reason, registering additional security providers or overriding existing ones should be done with caution.

Accelerating Class Loading

Class loading can be a time consuming process when many directories and classes are involved. Depending on your application and how it is used, the following server configuration parameters might help classes load more quickly.

`watt.server.coder.bincoder.trycontextloaderfirst`

When this property is set to true, when Integration Server encodes or decodes a pipeline, it uses the context loader before using the class loader for the currently executing thread. If a referenced class belongs to a particular package, it may be faster to use the context loader first.

`watt.server.classloader.pkgpriority=packageName, packageName`

Some Integration Server facilities do not operate within the context of a package. When a class is referenced from these facilities, all packages are searched in a non-deterministic order. This configuration parameter specifies a comma-delimited list of the packages that Integration Server is to search first. Specifying a package search order may offer a performance improvement. Refer to [“Server Configuration Parameters” on page 1017](#) for more information about this property.

51 Whitelist Filtering in Integration Server

■ About Whitelist Filtering in Integration Server	926
■ Enabling or Disabling Whitelist Class Filtering	926
■ Integration Server Whitelist Classes File	926
■ Creating a Package Whitelist Classes File	927
■ Logging Classes that Are Not on the Whitelist	928
■ Integration Server Blacklist Classes File	928

About Whitelist Filtering in Integration Server

Integration Server uses a whitelist filter to prevent the deserialization of unsafe Java objects. The Java deserialization vulnerability exists for applications that accept serialized objects but do not validate or check untrusted input before deserializing it properly. This gives attackers an opening to insert a malicious object into a data stream which could lead to remote code execution, denial of service, and many more.

In Integration Server the IDataBinCoder protocol used for the transmission of IData supports serialized Java objects. This feature could be exploited by configuring Integration Server to load third party libraries that can be used to construct a malicious serialized payload, which, in turn, can lead to arbitrary code execution and other serious security issues. If not addressed, this vulnerability can be exploited and damage the production environment.

To prevent Integration Server from deserializing untrusted Java objects, Integration Server includes a whitelist of classes that can be loaded and deserialized in Integration Server. When whitelist filtering is enabled, Integration Server deserializes a Java object only if the class appears in the whitelist. If Integration Server encounters a Java object whose class does not exist in the whitelist, Integration Server throws a `ClassNotFoundException`.

Integration Server includes a whitelist classes file for the server. You can also build a whitelist classes file for an individual package which can be used to identify custom classes used by the package.

Enabling or Disabling Whitelist Class Filtering

Whether or not Integration Server performs whitelist filtering is controlled by the server configuration parameter `watt.server.checkWhitelist`. When this parameter is set to true, whitelist filtering is enabled. Integration Server deserializes a Java object only if the class appears in the whitelist. If Integration Server encounters a Java object whose class does not exist in the whitelist, Integration Server throws a `ClassNotFoundException`. To enable or disable whitelist filtering, use Integration Server Administrator to change the value of the `watt.server.checkWhitelist` parameter.

Regardless of the `watt.server.checkWhitelist` value, Integration Server performs blacklist filtering when deserializing Java objects. For more information, see “[Integration Server Blacklist Classes File](#)” on page 928

Integration Server Whitelist Classes File

Software AG built a whitelist identifying classes used by Integration Server and layered products. The following file contains the list of whitelist classes:

Integration Server_directory /instances/*instanceName*/config/security/whitelistclasses.xml

If necessary, you can edit the `whitelistclasses.xml` file to include a custom class.

Note:

It is possible for an Integration Server fix or a subsequent release of Integration Server to change the contents of the `whitelistclasses.xml` file. If this occurs, any customizations to the

whitelistclasses.xml file will be lost. As an alternative, consider creating and using a whitelist for an individual package, described in [“Creating a Package Whitelist Classes File” on page 927](#).

➤ To edit the Integration Server whitelistclasses.xml file

1. Open the following file in text editor:

Integration Server_directory /instances/instanceName/config/security/whitelistclasses.xml

2. Add the following row to the classlist array: `<value>custom_class_name</value>`
3. Repeat the preceding step for each class that you want to add to the whitelist classes file.
4. Save your changes and close the file.
5. Restart Integration Server.

Creating a Package Whitelist Classes File

Integration Server supports the creation and use of a list of whitelist classes for an individual package. This can be useful when a custom package contains a Java service that deserializes a Java object. When the package loads, Integration Server adds the package’s whitelist classes to the allowed list of classes maintained in memory. When a package is disabled, Integration Server removes the package’s whitelist classes from the allowed list maintained in memory.

Note:

For Integration Server to use a package’s whitelist classes file, Integration Server must have whitelist filtering enabled. Integration Server performs whitelist filtering when the `watt.server.checkWhitelist` parameter is set to true.

➤ To create a package whitelist

1. Copy the Integration Server whitliestclasses.xml file located here:

Integration Server_directory /instances/instanceName/config/security/whitelistclasses.xml

2. Paste the copied whitelistclasses.xml file here:

Integration Server_directory /instances/instanceName/packages/packageName/config

3. Open the package whiteilstclasses.xml file in a text editor.
4. Delete all of the `<value>`elements
5. Add a row for each custom class you want to include in the package’s whitelist classes file.

`<value>custom_class_name</value>`

6. Save your changes to the package's `whitelistclasses.xml` file.
7. Reload the package for which you created the whitelist.

Logging Classes that Are Not on the Whitelist

You can configure Integration Server to generate a log file that lists any classes for which deserialization was attempted but are not part of a `whitelistclasses.xml` file. The log file can be useful for debugging. When `watt.server.dumpWhitelist` is set to true, Integration Server generates a log file that lists any classes for which deserialization was attempted but are not in a whitelist classes file. Specifically, Integration Server creates the following file during shut down:

Integration Server_directory /instances/*instanceName*/logs/runtimeClasslist.*time*.xml where *time* is the time in milliseconds since January 1, 1970.

For Integration Server to generate the `runtimeClasslist` file the `watt.server.checkWhitelist` and the `watt.server.dumpWhitelist` parameters must be set to true and an attempt to load a class not contained in the whitelist must have been made since Integration Server was last started.

Integration Server Blacklist Classes File

Integration Server performs blacklist class filtering during object deserialization. Integration Server detects the presence of unsafe classes during object deserialization and throws a `ClassNotFoundException` exception, which prevents the class from being instantiated.

Software AG built a blacklist file that is distributed with Integration Server. The blacklist is located here: *Integration Server_directory* instances/*instanceName*/config/security/classlist.xml

You can customize the blacklist classes file.

Note:

It is possible for an Integration Server fix or a subsequent release of Integration Server to change the contents of the blacklist file `classlist.xml`. If this occurs, any customizations to the `classes.xml` file will be lost. You will need to re-add your changes to the updated file.

> To edit the Integration Server blacklist classes file

1. Using a text editor, open the following file:
Integration Server_directory instances/*instanceName*/config/security/classlist.xml
2. Add the following row to the classlist array: `<value>custom_class_name</value>`
3. Repeat the preceding step for each class that you want to add to the blacklist classes file.
4. Save your changes and close the file.
5. Restart Integration Server.

52 Quiescing the Server for Maintenance

■ Overview	930
■ Specifying the Quiesce Settings	933
■ Quiescing Integration Server	933
■ Exiting Quiesce Mode	935
■ Deploying Quiesce Mode Configuration	936

Overview

Quiescing an Integration Server temporarily disables access to the server so you can perform maintenance tasks while the server is not connected to any external resources. In quiesce mode, any requests that are already in progress are permitted to finish, but any new inbound requests to the server are blocked. Outbound connection attempts, such as connections to JDBC pools or connections through LDAP or a central user directory, remain open.

What Happens when Integration Server Enters Quiesce Mode?

When Integration Server enters quiesce mode, the server disables or suspends the following assets and activities:

- **Ports.** Integration Server disables all ports except the diagnostic port and the port used to enter and exit quiesce mode.

For details about specifying a port to use for quiescing the server, see [“Specifying the Quiesce Settings” on page 933](#).

- **JMS messaging.** Integration Server disables all JMS connection aliases and suspends all JMS triggers including SOAP-JMS triggers and web service endpoint triggers. This prevents the retrieval and processing of new messages. Any trigger services that are running attempt to execute to completion. For more information about how Integration Server suspends JMS triggers, see [“Configuring Integration Server to Enable JMS Triggers Automatically after a Consumer Error” on page 865](#).
- **MQTT Messaging.** Integration Server disables all MQTT connection aliases and suspends all MQTT triggers. This prevents the retrieval and processing of new messages. Any trigger services that are running attempt to execute to completion. For more information about how Integration Server suspends MQTT triggers, see [Changing the State of an MQTT Trigger](#).
- **webMethods messaging.** Integration Server disables all webMethods messaging connection aliases that specify Universal Messaging as the messaging provider. Integration Server also suspends document retrieval and document processing for webMethods messaging triggers, which includes webMethods messaging triggers that receive documents from Universal Messaging, Broker, or locally published documents from the Integration Server. Any trigger services that are running attempt to execute to completion. For more information about how Integration Server suspends document retrieval for webMethods messaging triggers, see [“Suspending and Resuming Document Retrieval” on page 833](#). For more information about how Integration Server suspends document processing for webMethods messaging triggers, see [“Suspending and Resuming Document Processing” on page 842](#).
- **Packages.** Integration Server disables all packages except WmRoot and WmPublic. The server unloads the packages and their resources, completes the package’s shutdown services, and disables the packages in memory.

When you switch a server from active mode to quiesce mode, you can specify the number of minutes the server should wait before disabling packages. If services are running, this wait time gives the services a chance to complete before the server disables the package in which

they reside, or the packages in which their dependent services reside. If you do not specify a wait time, Integration Server disables packages immediately.

- **Scheduled user tasks.** Integration Server pauses the scheduler, permits the completion of any scheduled user tasks that are already running, and suspends any scheduled user tasks that have not yet started. If a running user task does not complete before its dependent services are disabled, the task will end in error because of the unresolved dependencies.
- **Guaranteed delivery.** Integration Server shuts down the inbound and outbound guaranteed delivery Job Managers and stops both inbound and outbound guaranteed delivery of documents. For outbound delivery, the server sets the watt property `watt.tx.disabled` to `True` to disable the use of guaranteed delivery for outbound requests.

Note:

Because the server forcefully shuts down the inbound and outbound Job Managers when entering quiesce mode, in-process transactions may fail. Avoid submitting any transactions for processing as the server enters quiesce mode.

If you want to receive inbound guaranteed delivery requests while the server is in quiesce mode, you must first make sure the watt property `watt.server.tx.heuristicFailRetry` is set to `True`. Then, start the inbound Job Manager using the service `pub.tx:init`. The server will re-execute the services for transactions in the job store that are in `PENDING` status.

If you want to submit outbound transactions while the server is in quiesce mode, you must first set the watt property `watt.tx.disabled` to `False`. Then, start the outbound Job Manager using the service `pub.tx:resetOutbound`.

- **Clustering.** Integration Server shuts down all cache managers in the *Integration Server_directory* \instances\instance_name\config\Caching directory and exits the cluster.

The following activities continue when the server is in quiesce mode:

- **Active tasks.** Integration Server completes *user tasks* that were started before Integration Server entered quiesce mode. However, if the **Disable System Tasks interacting with Database** option on the **Settings > Quiesce** page is selected, then the Database-related system tasks are also disabled.
- **Audit logging.** By default, the server continues to write audit logging records to the database in the quiesce mode. However, if the **Disable Audit Logging** option on the **Settings > Quiesce** page is selected, then audit logging is also disabled.
- **Enterprise Gateway.** If the Integration Server acting as the Internal Server is quiesced, the server disconnects any existing connections to the Enterprise Gateway Server and completes user tasks that were started before the server entered quiesce mode. If the Integration Server acting as the Enterprise Gateway Server is quiesced, the server's external and registration ports are disabled, any existing connections to the Internal Server are disconnected, and any requests made by the Internal Server to the Enterprise Gateway Server will time out.

In an Enterprise Gateway scenario, you can avoid unnecessary connection attempts between the two servers by doing one of the following:

- Quiesce the Enterprise Gateway Server and the Internal Server at the same time.

- Disable the Enterprise Gateway registration port on the server that is not being put into quiesce mode.

For more information about webMethods Enterprise Gateway, see [“Configuring webMethods Enterprise Gateway” on page 557](#).

What Tasks Can You Perform in Quiesce Mode?

While Integration Server is in quiesce mode, you can:

- Create, modify, and deploy packages.
- Change server settings.
- Change the state of ports, JMS connection aliases, webMethods messaging connection aliases, packages, and scheduled user tasks.
- Change clustering configuration such as modifying the Terracotta Server Array URLs.
- Update cache and cache manager settings.
- Install fixes and perform other maintenance tasks.
- Debug services.

What Happens when Integration Server Exits Quiesce Mode?

When Integration Server exits quiesce mode, the server:

- Retains state changes made to any assets during the quiesce session. For example, if you enabled a package that was disabled before the server entered quiesce mode, the package remains enabled when the server exits quiesce mode. If you created a package and then disabled it, the package remains disabled when the server exits quiesce mode.
- Restores the state of any unchanged assets to their state before the server entered quiesce mode. For example, if a scheduled user task was suspended before the server was quiesced, the task remains suspended when the server exits quiesce mode.
- Enables any feature that you disabled on the **Settings > Quiesce** page.
- Resumes the task scheduler.
- Resumes clustering activities.

If Integration Server encounters any errors that prevent it from joining the cluster when exiting quiesce mode, Integration Server does not re-enter quiesce mode. Instead, Integration Server runs as an unclustered, stand-alone server. Integration Server logs any errors that occurred while attempting to join the cluster in the report that appears in Integration Server Administrator after exiting quiesce mode. Use the report and error logs to help correct the underlying configuration errors. Then, to rejoin the cluster, either restart Integration Server or enter and then exit quiesce mode.

- Starts the guaranteed delivery inbound and outbound Job Managers, if you did not start them while the server was in quiesce mode. The server also sets the `watt.tx.disabled` property to `False` and resumes inbound and outbound guaranteed delivery of documents.

Specifying the Quiesce Settings

Before you can start or restart an Integration Server in quiesce mode for the first time, you must specify the port through which the server will enter and exit quiesce mode. This port and the diagnostic port are the only ports that remain enabled when a server enters quiesce mode. You cannot disable the quiesce port or the diagnostic port when the server is in quiesce mode.

Note:

The quiesce port must reside in either the `WmRoot` or the `WmPublic` package. The port must also be enabled and have “allow” or “allow+” privilege. If the quiesce port is not specified, is suspended or disabled, or does not have “allow” or “allow+” privileges, Integration Server will not be able to enter quiesce mode.

➤ To specify the quiesce port

1. Open the Integration Server Administrator if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Quiesce**.
3. From the **Select New Quiesce Port** list, select the port you want Integration Server to use as the quiesce port.

Note:

This list does not include the diagnostic port or any HTTP/HTTPS ports created for Internal Servers in a `webMethods Enterprise Gateway` configuration.

4. Click **Save Changes**.

Quiescing Integration Server

You can quiesce Integration Server by starting the server in quiesce mode from the command prompt, by switching a server from active mode to quiesce mode from Integration Server Administrator, or by restarting a server in quiesce mode from Integration Server Administrator.

Additionally, an Integration Server that is a member of a cluster can be configured to enter quiesce mode if any errors prevent the server from joining the cluster at start up. To do this, when configuring a cluster, set the **Action on Startup Error** to **Enter Quiesce Mode on a Stand-Alone Integration Server**. For more information about entering quiesce mode automatically when Integration Server cannot connect to the cluster at start up, see the *webMethods Integration Server Clustering Guide*.

Note:

Before you can start an Integration Server in quiesce mode for the first time, you must specify the port through which the server will enter and exit quiesce mode. For more information, see “Specifying the Quiesce Settings” on page 933.

Starting the Server in Quiesce Mode from the Command Prompt

You can start the server in quiesce mode by setting the `-quiesce` switch. For information about other startup command switches, see “Starting a Server Instance from the Command Prompt” on page 39.

Quiescing an Active Server from Integration Server Administrator

You can switch an active Integration Server to quiesce mode using the Integration Server Administrator.

➤ To quiesce an active server from Integration Server Administrator

1. Open the Integration Server Administrator if it is not already open.
2. In the upper right corner of any Integration Server Administrator screen, click **Enter Quiesce Mode**.

Note:

The **Enter Quiesce Mode** link is available only when Integration Server is in active mode.

3. When you are prompted to confirm that you want the server to switch to quiesce mode, do the following:
 - a. If services are running and you want them to complete before the server starts disabling packages, specify the number of minutes the server should wait. The default is 0, which means the server will disable packages immediately.
 - b. Click **OK**.
4. If you are running Integration Server Administrator from a port other than the quiesce port, stop Integration Server Administrator and start it using the quiesce port.

Integration Server displays a message at the top of all pages in the Integration Server Administrator indicating that the server is running in quiesce mode.

Note:

If the server cannot disable or suspend an asset or activity as it enters quiesce mode, the server displays a warning message and continues the quiesce process. If the condition stated in the warning interferes with a maintenance task you intend to perform, resolve the issue stated in the warning and then restart the server in quiesce mode.

Restarting a Server in Quiesce Mode from Integration Server Administrator

You can restart an active or quiesced Integration Server in quiesce mode from the Integration Server Administrator.

➤ To restart a server in quiesce mode from Integration Server Administrator

1. Open the Integration Server Administrator if it is not already open.
2. In the upper right corner of any Integration Server Administrator screen, click **Shut Down and Restart**.
3. On the Shut Down and Restart screen, select whether you want the server to wait before restarting or to restart immediately:

After all client sessions end. Specify the number of minutes you want the server to wait before restarting. The server begins monitoring user activity and automatically restarts in quiesce mode when all non-administrator sessions complete, or when the time you specify elapses, whichever comes first.

Immediately. The server and all active sessions terminate immediately. The server then restarts in quiesce mode.

4. Click **Restart in Quiesce Mode**.
5. When you are prompted to confirm that you want to restart the server, click **OK**.
6. If you are running Integration Server Administrator from a port other than the quiesce port, stop Integration Server Administrator and start it using the quiesce port.

Integration Server displays a message at the top of all pages in the Integration Server Administrator indicating that the server is running in quiesce mode.

Note:

If the server cannot disable or suspend an asset or activity as it enters quiesce mode, the server displays a warning message and continues the quiesce process. If the condition stated in the warning interferes with a maintenance task you intend to perform, resolve the issue stated in the warning and then restart the server in quiesce mode.

Exiting Quiesce Mode

When an Integration Server is running in quiesce mode, you can use the Integration Server Administrator to exit quiesce mode and resume normal operation.

➤ To exit quiesce mode

1. Open the Integration Server Administrator if it is not already open.
2. In the upper right corner of any Integration Server Administrator screen, click **Exit Quiesce Mode**.

Note:

The **Exit Quiesce Mode** link is available only when Integration Server is in quiesce mode.

3. When you are prompted to confirm that you want the server to exit quiesce mode, click **OK**.

After exiting quiesce mode, Integration Server Administrator displays a report that indicates the state of various operations at the time the server exited quiesce mode.

Deploying Quiesce Mode Configuration

When you specify a quiesce port, Integration Server maintains the port alias in the quiesce configuration file (quiesce.cnf). This file resides in the *Integration Server_directory* \instances*instance_name*\config directory. The quiesce.cnf file is a deployable asset.

When you use webMethods Deployer to deploy the quiesce.cnf file to another Integration Server, keep the following points in mind:

- The quiesce port you want to deploy must reside in either the WmRoot or the WmPublic package. Because all other packages are disabled when a server enters quiesce mode, the target server will not be able to enter quiesce mode if the quiesce port resides in a package other than WmRoot or WmPublic.
- Packages that are installed with Integration Server, such as WmRoot and WmPublic, are not deployable assets. Deployer identifies these packages as unresolved dependencies when you include the quiesce.cnf file as part of the deployment set. Click **Ignore** to bypass dependency checking for this situation.
- If no quiesce port is specified on the target server, the target server uses the port specified in quiesce.cnf. Either ensure that this port exists on the target server or add the port to the list of deployable assets. The port should be of type HTTP or HTTPS.
- You can substitute the quiesce port alias value during the mapping step in Deployer.

For details about deploying assets using Deployer, see *webMethods Deployer User's Guide*.

53 Diagnosing the Integration Server

■ Introduction	938
■ Configuring the Diagnostic Port	938
■ Using the Diagnostic Utility	939
■ Starting the Integration Server in Safe Mode	942
■ When the Server Automatically Places You in Safe Mode	943
■ Generating Thread Dumps	943

Introduction

This chapter contains information for the server administrator who troubleshoots the Integration Server or maintains diagnostic data from the server. Diagnostic data is the configurational, operational, and logging information from the Integration Server. This information is useful in situations where the server becomes unresponsive and unrecoverable.

To facilitate the troubleshooting process, the Integration Server provides the following features:

- **Diagnostic port.** A special port that uses a dedicated thread pool.
- **Diagnostic utility.** A special service that extracts important diagnostic data from the Integration Server.
- **Safe mode switch.** A method of starting the Integration Server in which the server does not connect to any external resource.
- **Thread dump.** A facility to generate a log containing information about currently running threads and processes within Java Virtual Machine (JVM), to help diagnose issues with Integration Server.

Configuring the Diagnostic Port

The diagnostic port is a special port that uses threads from a dedicated thread pool to process requests submitted via HTTP. It behaves like a typical HTTP port, except that the server uses the diagnostic thread pool instead of the server thread pool.

By maintaining a separate thread pool, this port improves the troubleshooting capability when the server becomes unresponsive. For example, when the server reaches its maximum number of threads, you cannot open the Integration Server Administrator. This prevents you from accessing information that might help you determine why the threads are not available. It also prevents you from freeing up other server resources. Using the threads from the diagnostic thread pool, the diagnostic port enables you to open the Integration Server Administrator.

When you install the Integration Server, it automatically creates the diagnostic port at 9999. If another port is running at 9999, the server will disable the diagnostic port when you start the Integration Server. To enable the diagnostic port, you must edit the port number. For instructions about how to edit port configurations, see [“Editing a Port” on page 206](#). Only one diagnostic port can exist on each Integration Server.

Diagnostic Thread Pool Configuration

Through the Integration Server Administrator, you can configure the number of threads in the diagnostic thread pool. The server adds threads to the pool as needed until it reaches the maximum allowed. If the server reaches the maximum number, it waits until processes complete and returns threads to the pool before beginning more processes.

You can also set the thread priority for the diagnostic thread pool. The diagnostic thread priority determines the order of execution when the JVM receives requests from different threads. The larger the number, the higher the priority. When the JVM receives requests from different threads,

it will run the thread with the higher priority. Therefore, by assigning a higher priority to the threads in the diagnostic thread pool, you can take advantage of the dedicated thread pool and improve access to the Integration Server Administrator.

For more information about how to configure the diagnostic thread pool, see [“Working with Extended Configuration Settings” on page 127](#).

Diagnostic Port Access

Only users in the Administrators group can access the server through the diagnostic port. You can access the Integration Server Administrator via `http://<hostname>:<diagnostic port>` where *hostname* is the machine that hosts the Integration Server and *diagnostic port* is the diagnostic port number. After prompting you for a username and password, the server displays the Integration Server Administrator. Because you can access the diagnostic port only through HTTP, data and passwords will be passed clear=unencrypted.

The diagnostic port allows access only to services defined with the Administrators ACL. Software AG recommends that you do not change the default access settings.

Note:Software AG strongly recommends that you discourage any external user access to the diagnostic port and utility. LDAP users should not access the diagnostic port.

Using the Diagnostic Utility

You use the diagnostic utility to collect configuration, operation, and logging data from the Integration Server. You can also use the diagnostic utility to view the list of fixes applied to the installed packages and Integration Server. The diagnostic utility is an Integration Server service called `wm.server.admin:getDiagnosticData`. It is accessible only by members of the Administrators group. Although you run the utility via the diagnostic port to troubleshoot, it can also be used with any HTTP or HTTPS port to collect diagnostic data periodically. You can access the service through the **Get Diagnostics** button in Integration Server Administrator.

The diagnostic utility creates a temporary `diagnostics_<hostname>_<port>_<yyyyMMddHHmmss>.zip` file in the `Integration Server_directory \instances\instance_name\logs` directory and writes to the .zip file as it collects information. It also contains a `config\PackagesAndUpdates.txt` file, which lists the packages and package updates for the Integration Server.

If there are problems creating the .zip file, such as insufficient space in the file system, it will return a text file. In the text file, the configuration and operation data are separated into distinct sections for easier reading. Unlike the .zip file, the text file does not contain logging data.

The .zip file contains a file in the config directory called `PackagesAndUpdates.txt`. This file lists the packages and package updates for the Integration Server.

Diagnostic Utility Performance

The diagnostic utility can execute slowly when logging large amounts of data from the Integration Server. To increase performance, you can set limits to the amount of data the diagnostic tool returns by specifying a `maxLogSize` value or setting the `watt.server.diagnostic.logperiod` parameter.

The `maxLogSize` parameter of the `wm.server.admin:getDiagnosticData` service sets the size limit for log files written to the `diagnostic_data.zip` file. If a log file exceeds the specified `maxLogSize`, the diagnostic utility omits it from the `.zip` file but records it in a `diagnosticwarning.txt` file. This file lists all of the log files that exceed the `maxLogSize` value. It is located in the `logs` directory of the `.zip` file.

Note:

You can use the `maxLogSize` parameter only when running the diagnostic utility from a browser. You cannot limit the log size when you run the diagnostic utility from Integration Server Administrator. For more information, see [“Running the Diagnostic Utility Service” on page 941](#).

Use the `watt.server.diagnostic.logperiod` parameter to specify the log period. By default, it is set to 6 hours. When this property is set to 0, the utility does not return any log files. It returns only the configuration and run-time data files. The logging information the utility returns depends on how you store the logs. If you save the logs to a database, the diagnostic utility will return the exact number of log entries that match the specified number of hours. If you save the logs to the file system, it will return not only the period within the specified number of hours but the entire log for that day. For instructions about how to set server configuration parameters, see [“Working with Extended Configuration Settings” on page 127](#).

Use the `watt.server.diagnostic.logFiles.maxMB` parameter to specify the size limit for including audit log tiles in the diagnostic archive. While collecting each audit log file, Integration Server calculates the total size of the log files for the requested log period. If the total size of the log files for a particular audit log exceeds the value of `watt.server.diagnostic.logFiles.maxMB` for the log period, Integration Server does not include that audit log file in the diagnostic archive. Consider the examples below.

Example 1

- `watt.server.diagnostic.logFiles.maxMB` is 250 and `watt.server.diagnostic.logperiod` is 6.
- There are two `WMSESSION` log files that cover the previous six hours.
- The total size of the two `WMSESSION` log files is greater than 250 MB.

RESULT: No session audit log data will be included in the diagnostic data archive.

Example 2

- `watt.server.diagnostic.logFiles.maxMB` is 300 and `watt.server.diagnostic.logperiod` is 8.
- There is one `WMSERVICE` log file that covers the previous eight hours.
- The size of the `WMSERVICE` log file is less than 300 MB.

RESULT: Service audit log data will be included in the diagnostic data archive.

Running the Diagnostic Utility from Integration Server Administrator

Complete the following procedure to run the diagnostic utility from Integration Server Administrator.

➤ **To run the diagnostic utility from Integration Server Administrator**

1. Open Integration Server Administrator if it is not already open.
2. Click **About** in the upper right-hand part of the screen.
3. In the **Software** area, click **Get Diagnostics**.
4. You can choose to perform one of the following:
 - a. **Open** the diagnostic data file.
 - b. **Save** the diagnostic data file to the client file system.
 - c. **Cancel** and exit this operation.

Note:

If you save or open the diagnostic data file, it opens or saves the file to the *client* system. Integration Server automatically saves a copy to the *Integration Server_directory \instances\instance_name\logs* directory of the host machine where Integration Server is running.

Running the Diagnostic Utility Service

Complete the following procedure to run the diagnostic utility without using Integration Server Administrator. For example, you would use this method if you wanted to use the `maxLogSize` parameter to limit the size of the `.zip` file.

➤ **To run the diagnostic utility without using Integration Server Administrator**

1. Start your web browser.
2. Type the following URL:

```
http://<hostname>:<port>/invoke/wm.server.admin/getDiagnosticData
```

where *<hostname>* is the IP address or name of the machine and *<port>* is the port number where the Integration Server is running.

Note:

You can limit the byte size of the log files included in the `.zip` file by adding the `maxLogSize` parameter to the URL as follows:

```
http://<hostname>:<port>/invoke/wm.server.admin/getDiagnosticData?maxLogSize
=number_of_bytes
```

3. Log on to the Integration Server with a username and password that has administrator privileges.
4. You can choose to perform one of the following:
 - a. **Open** the diagnostic data file.
 - b. **Save** the diagnostic data file to the file system.
 - c. **Cancel** and exit this operation.

Note:

If you save or open the diagnostic data file, it opens or saves the file to the *client* system. Integration Server automatically saves a copy to the *Integration Server_directory* \instances*instance_name*\logs directory of the host machine where Integration Server is running.

Starting the Integration Server in Safe Mode

If Integration Server is having trouble starting because it or one of its packages cannot connect to an external resource, you can stop Integration Server and then start it in *safe mode*. When you start Integration Server in safe mode, it does not connect to any external resources, including databases. As a result, when Integration Server is in safe mode, it writes audit logging data associated with the IS Core Audit and Process Audit functions to flat files on the Integration Server instead of to an external database. In addition, when in safe mode, Integration Server loads only the WmRoot package; all other packages are inactive. When you restart Integration Server after you diagnose and correct the problem, Integration Server resumes audit logging for IS Core Audit and Process Audit functions to the external database and loads all enabled packages.

Important:

Use safe mode for diagnostic or troubleshooting purposes only. Do not run any regular Integration Server tasks or Designer while in safe mode. It will return unpredictable results.

If Integration Server could not connect to a Broker or database, check the appropriate connection parameters and modify them as necessary. For instructions, see the *webMethods Audit Logging Guide*.

If a package such as Trading Networks Server or the webMethods SAP Adapter could not connect to an external resource, open Integration Server Administrator and go to the **Packages > Management > Activate Inactive Packages** page. In the **Inactive Packages** list, select the package and click **Activate Package**. Integration Server puts the package into the state it would have been in if you had started Integration Server normally. For example, if the package would have been enabled, Integration Server loads and enables it. Check and modify the connection parameters using the instructions in the appropriate guide.

Starting Integration Server in Safe Mode

> To start Integration Server in safe mode

1. Stop the Java process associated with the Integration Server (for example, in Windows Task Manager).
2. At the command line, go to the home directory of the server instance (Integration Server\profiles\IS_*instance_name*) and enter one of the following commands to start the server.

System	Command
Windows	bin\console.bat -safeboot (other switches)
UNIX	bin/console.sh -safeboot (other switches)

For information about other switches, see [“Starting a Server Instance from the Command Prompt” on page 39](#). When you open the Integration Server Administrator, it will display a message indicating that the server is running in safe mode.

When the Server Automatically Places You in Safe Mode

If the Integration Server detects a problem with the master password or outbound passwords at startup, it will automatically place you in *safe mode*. You will see the following message in the upper left corner of the Server Statistics screen of the Integration Server Administrator:

```
SERVER IS RUNNING IN SAFE MODE. Master password sanity check failed -- invalid
master password provided.
```

These problems can be caused by a corrupted master password file, a corrupted outbound password file, or by simply mis-typing the master password when you are prompted for it. If you suspect you have mis-typed the password, shut down the server and restart it, this time entering the correct password. If this does not correct the problem, refer to [“When Problems Exist with the Master Password or Outbound Passwords at Startup” on page 546](#) for instructions.

Generating Thread Dumps

If Integration Server or a subsystem becomes slow or unresponsive, or users are unable to log into Integration Server, you can generate thread dumps to help you diagnose the problem. A thread dump can help you locate thread contention issues that can cause thread blocks or deadlocks.

You can generate thread dumps of the following:

- The JVM in which the Integration Server is running
- Individual threads running on Integration Server

Based on the information you obtain from these thread dumps, you might be able to correct the problem.

If you detect a problem with a thread that is associated with a user-written Java service or a flow service, you have the option of canceling or killing the thread.

When you cancel a thread, Integration Server frees up resources that are held by the thread and returns the thread to the thread pool. If Integration Server cannot cancel the thread, it gives you the option of killing the thread. When you kill a thread, Integration Server terminates the thread and adds a new one to the thread pool. For more information about canceling and killing service threads, see [“Canceling and Killing Threads Associated with a Service” on page 708](#).

The following example describe how you might use the JVM thread dump and individual thread dumps to diagnose and fix problems.

Scenario 1: A Service Is Running Longer than Expected

1. A Flow service has been running for a very long time. You suspect the service is in an infinite loop, or is waiting for external resources that are not available.
 2. You log in through the Integration Server primary port and navigate to the **Statistics > System Threads** screen.
 3. On the **System Threads** screen you see threads that are associated with the service in question. You look at individual dumps of those threads.
 4. Using the information provided in the dumps, you determine that the threads are experiencing contention issues.
 5. You cancel the threads. This action allows the service to complete.
-

Scenario 2: The Server Is Unresponsive, Users Cannot Log In Through the Primary Port

1. Integration Server is unresponsive and no one can log in through the primary port.
 2. You log in through the diagnostic port and navigate to the **Statistics > System Threads** screen.
 3. On the **System Threads** screen you see multiple threads for the same service. You look at individual dumps of those threads.
 4. Based on the information provided in the dumps, you try canceling the threads. The problem persists.
 5. You try killing the threads. The problem persists.
 6. You perform a JVM dump.
 7. Using the information provided in the JVM dump, you determine the cause of the problem and are able to resolve it.
-

The following procedures show how to generate dumps for individual threads and for the JVM.

Generating a Dump of an Individual Thread

> To view information about an individual thread

1. Open the Integration Server Administrator if it is not already open.
2. From the **Server** menu in the Navigation panel, click **Statistics**.
3. In the **System Threads** field, in the **Current** column, click the number of current threads.

The **System Threads** screen displays. This screen contains a list of all the threads that are currently running on the server.

4. To view a dump of a particular thread, in the **Name** column for that thread, click the thread name.

The server displays a dump of that thread.

Generating a Dump of the JVM

> To generate a JVM thread dump

1. Open the Integration Server Administrator if it is not already open.
2. From the **Server** menu in the Navigation panel, click **Statistics**.
3. Under **Usage**, click the **Current** number of **System Threads**.

The **Server > Statistics > System Threads** page is displayed.

The **System Threads** table lists thread names and displays information about them.

4. Click **Generate JVM Thread Dump**.

The **Server > Statistics > System Threads > Thread Dump** page is displayed, showing the dump.

5. Click **Return to System Threads** to return to the **System Threads** page.
6. For information about server recovery when a hardware or software problem forces you to do a server restart, see [“Starting and Stopping the Server” on page 37](#).

54 Debugging Service Exceptions Using the Error Log

■ Introduction	948
■ Controlling the Level of Exception Logging Detail	948
■ Displaying the Error Log	948
■ Interpreting the Error Log	948

Introduction

The Integration Server error log maintains information about exceptions that are thrown by services. This appendix describes how to use this log to help debug service exceptions.

Controlling the Level of Exception Logging Detail

You can control how the Integration Server logs service exceptions, and to what level of detail, by setting the `watt.server.deprecatedExceptionLogging` parameter. The information in this appendix assumes that this parameter is set to its default setting, `false`, which provides more detailed exception logging. For more information about this parameter, see [“Server Configuration Parameters” on page 1017](#).

Displaying the Error Log

In Integration Server Administrator, go to the **Logs > Error** page to view the error log. The fields in the error log are listed below.

Column	Details
Timestamp	Date and time the entry was written to the log.
Service Name	Name of the service in which the exception occurred.
Service Stack	Parent services for the service in which the exception occurred.
Error Message	Message that describes the exception that occurred.
Stack Trace	Trace that shows the call sequence leading to the exception. To expand the display of stack trace data, select the Expand Stack Trace Data check box in the Log display controls area and click Refresh .
Root Context	Context information webMethods Monitor uses to connect related entries from different logs.
Parent Context	
Current Context	

Interpreting the Error Log

The following image shows a portion of the error log that includes details that help you identify which step caused an exception. The Integration Server Administrator displays these details when you perform two actions:

- Set the value of the `watt.server.deprecatedExceptionLogging` parameter to its default setting, `false`.
- Select the **Expand Stack Trace Data** check box on the **Logs > Error** page.

Integration Server error log showing detailed exception logging.

Time Stamp	Service Name	Service Stack	Error Message	Stack Trace
End of Current Log				
2016-03-02 23:11:08 EST	test:flowMiddle	test:exInner test:exMiddle test:flowWithoutCatch(0) test:flowMiddle(2/0/0)	com.wm.app.b2b.server.ServiceException: testing inner Caused by: com.wm.app.b2b.server.ServiceException: testing inner	com.wm.app.b2b.server.ServiceException: testing inner at test:exInner(test.java:35) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethod) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.lang.reflect.Method.invoke(Method.java:497) at com.wm.app.b2b.server.JavaServiceBase.invoke(JavaService.java:406) at com.wm.app.b2b.server.InvokeManager.process(InvokeManager.java:549) at com.wm.app.b2b.server.UTISpace.ReservationProcessor.process(ReservationProcessor.java:39) at com.wm.app.b2b.server.Invoke.StatisticProcessor.process(StatisticProcessor.java:49) at com.wm.app.b2b.server.Invoke.ServiceCompletionImpl.process(ServiceCompletionImpl.java:243) at com.wm.app.b2b.server.Invoke.ValidateProcessor.process(ValidateProcessor.java:49) at com.wm.app.b2b.server.Invoke.PipelineProcessor.process(PipelineProcessor.java:171) at com.wm.app.b2b.server.ACLManager.process(ACLManager.java:303) at com.wm.app.b2b.server.Invoke.DispatchProcessor.process(DispatchProcessor.java:34) at com.wm.app.b2b.server.AuditLogManager.process(AuditLogManager.java:377) at com.wm.app.b2b.server.Invoke.InvokeManager.invoke(InvokeManager.java:548) at com.wm.app.b2b.server.Invoke.InvokeManager.invoke(InvokeManager.java:385) at com.wm.app.b2b.server.ServiceManager.invoke(ServiceManager.java:238) at com.wm.app.b2b.server.ServiceManager.invoke(ServiceManager.java:107) at com.wm.app.b2b.server.ServiceManager.invoke(ServiceManager.java:79) at com.wm.app.b2b.server.Service.doInvoke(Service.java:62) at com.wm.app.b2b.server.Service.doInvoke(Service.java:55) at test:exMiddle(test.java:50) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethod) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.lang.r...truncated-

List of parent services and index identifying step that caused the exception

Concatenated messages from each nested exception

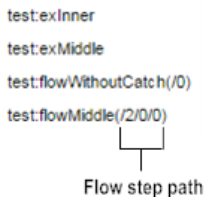
Innermost stack trace

Understanding the Service Stack

The **Service Stack** column shows the Java services and flow services called up to the point at which the exception occurred. The stack lists services in the following order:

- test:exInner ————— Service executing when the exception occurred
- test:exMiddle
- test:flowWithoutCatch(0)
- test:flowMiddle(2/0/0) ————— First service executed

For flow services, the **Service Stack** column also shows an indexed path that identifies the step within a service at which the exception occurred. The path takes the form `/n/n/n...`, where `n` represents a sequential index of flow steps and `/` indicates a level of nesting:



This path helps pinpoint the step at which the exception occurred and is particularly useful when a flow service is called multiple times or when the service or step is deeply nested. For example, suppose `serviceA` invokes `serviceB` three times, and the exception occurred at the second invocation of `serviceB`. The **Service Stack** column in the error log would show the path as `serviceA(/1/2)`:

Step	Index
INVOKE serviceB	/0

Step	Index
SEQUENCE	/1
INVOKE serviceD	/1/0
INVOKE serviceE	/1/1
INVOKE serviceB	/1/2 (exception thrown here)
SEQUENCE	/2
INVOKE serviceC	/2/0
INVOKE serviceB	/2/1

Understanding the Error Messages

When you specify detailed exception logging, the error log's **Error Message** column shows the error message that corresponds to the exception and a concatenation of the error messages from each of the nested exceptions. The format of these messages is as follows:

```
Outer exception message
Caused by: middle.exception.className: Middle exception message
Caused by: inner.exception.className: Inner exception message
```

In addition to the standard error message text that Integration Server provides (for example, "[ISS.0062.9021] "object" is null"), exception messages may also contain additional, customized text that a service developer provides in the error handling portion of the service.

Understanding the Stack Trace

Exceptions often contain other exceptions nested within them. The extent to which the **Stack Trace** column reports the call sequence leading to the exceptions depends on the setting you select for the `watt.server.deprecatedExceptionLogging` parameter, as follows:

- If this parameter is set to `false` (detailed exception logging), the error log's **Stack Trace** column shows the innermost stack trace (that is, the stack trace that points to the source of the problem).
- If this parameter is set to `true` (basic exception logging), the stack trace is often truncated and the cause of the exception becomes more difficult to trace. For this reason, Software AG recommends that you do not set this parameter to `true` unless you are executing services that catch exceptions and do not rethrow them.

55 Using Integration Server with Docker

■ Overview of Docker and Integration Server	952
■ About the is_container Script	953
■ Prerequisites for Building a Docker Image	954
■ Specifying Services to Expose to Consumers in webMethods Cloud	955
■ Building the Docker Image for an Integration Server Instance	955
■ Loading a Docker Image to an On-Premise Docker Registry	961
■ Pushing a Docker Image to an On-Premise Docker Registry	962
■ Running the Docker Image in an On-Premise Docker Container	963
■ Pushing the Docker Image to Integration Cloud	969
■ Running the Docker Image in Integration Cloud	970
■ Stopping a Docker Container for Integration Server	970
■ Using a Configuration Variables Template with a Docker Image	970

Overview of Docker and Integration Server

Docker is an open-source technology that allows users to deploy applications to software containers. A Docker container is an instance of a Docker image, where the Docker image is the application, including the file system and runtime parameters. You can create a Docker image from an installed and configured Integration Server instance and then run the Docker image inside a Docker container. To facilitate running Integration Server in a Docker container, Integration Server provides a script to use to build a Docker image and then load or push the resulting Docker image to a Docker registry hosted on-premise or in webMethods Integration Cloud.

Note:

You can build a docker image for instances of Integration Server, Microservices Runtime, or Integration Agent. Unless otherwise specified, the instructions for working with Integration Server and Docker apply to Microservices Runtime and Integration Agent as well.

Support for Integration Server with Docker 1.12.11 and later is available on the following where Docker provides native support: Linux systems, UNIX systems, and Windows Server 2016.

Note:

The Integration Server documentation assumes a familiarity with Docker technology. An in-depth discussion of Docker and container technology is beyond the scope of this guide but is available elsewhere.

Recommendations for Using Integration Server with Docker

If you opt to run Integration Server in a Docker container, Software AG recommends the following:

- Create a Docker image for an installed, fully configured on-premise Integration Server. Make sure the server configuration is as complete as possible before creating the image. While some configurations can be changed or added after the Docker image is created, it is best configure as much as possible in the base image.
- Microservices Runtime and Integration Server provide features that you can use to add or change some of the configuration or content of a server running in a Docker container without needing to recreate the Docker image, specifically:
 - Configuration variables templates can be used to change some server configuration for an Integration Server or Microservices Runtime that runs in a Docker container. Configuration variables templates can also be used to add assets to an Integration Server or Microservices Runtime in a Docker container and set asset properties that were previously not set.

Software AG recommends adding configuration in the base Docker image. Use configuration variables to change endpoint or configuration values across the environment.

For more information about configuration variables, see *Developing Microservices with webMethods Microservices Runtime*.

- The EXTERNALIZE_PACKAGES ENV variable can be used to make more packages available for use with an Integration Server that runs in a Docker container. You can specify the EXTERNALIZE_PACKAGES ENV variable when you start the Docker container. For

more information about this environment variable, see [“Environment Variables for Use with Docker”](#) on page 1183.

- Automatic package deployment enables the installation or update of custom packages on a server running in a Docker container without needing to rebuild the Docker image. For more information about automatic package deployment, see *Developing Microservices with webMethods Microservices Runtime*.

If you do not use the features described above, consider a Docker image of Integration Server to be immutable. That is, do not make configuration or content changes on an Integration Server running in Docker container. Instead, make any changes on the on-premise Integration Server, recreate the Docker image, load or push the Docker image to the Docker repository, and then start a Docker container for the image.

Note:

The configuration variables template and automatic package deployment feature are available with Microservices Runtime or an Integration Server with additional licensing.

- If you intend to run a Docker image created for Integration Server in Integration Cloud, Software AG strongly recommends that you enable and configure CSRF guard in the Integration Server instance for which you are creating a Docker image. Enabling the CSRF guard feature prevents Cross-Site Forgery Request (CSRF) attacks through the Integration Server Administrator URLs. Enable and configure CSRF guard on the Integration Server instance before creating a Docker image for the instance. For information about enabling and configuring CSRF guard, see [“Securing Integration Server with CSRF Guard”](#) on page 551.

About the is_container Script

The is_container.sh/bat script facilitates interaction with Docker. The script provides commands for creating a Dockerfile, building Docker images for Integration Server, loading or pushing the Docker image to a Docker registry hosted on-premise, and pushing the Docker image to a Docker registry hosted on webMethods Cloud.

The intent of the is_container.sh/bat script is to provide the following:

- A way for customers to take advantage of the webMethods Integration Cloud Container Edition. The script works seamlessly with webMethods Integration Cloud.
- A starter script for users of on-premise Integration Server who want to use Docker in their infrastructure.

Note:

The is_container.sh/bat script is intended as a starter script. The script does not provide support for all Docker capabilities. Software AG anticipates that users will use the is_container.sh/bat script as a sample to reference when creating their own scripts.

The is_container.sh/bat script is located in *Software AG_directory\ Integration Server_directory \docker*

is_container Script Commands

The following table provides descriptions of for the commands that an be used with the `is_container.sh/bat` script.

Command	Description
<code>createDockerfile</code>	Creates a Dockerfile for a base Integration Server instance, including "Default" and "Wm" packages only.
<code>createLeanDockerfile</code>	Creates a Dockerfile for a base Integration Server instance, including the "Default" package and the "Wm" packages that are required for core Integration Server such as WmRoot, WmPublic, and WmCloud.
<code>createPackageDockerfile</code>	Creates a Dockerfile for custom packages.
<code>build</code>	Executes Docker build using the provided Dockerfile to build a base image of Integration Server
<code>buildPackage</code>	Executes Docker build using the provided Dockerfile to build an image of Integration Server that contains custom packages.
<code>saveImage</code>	Saves the image from the local Docker registry into a tar file specified by the <code>file.path</code> parameter.
<code>loadImage</code>	Loads the image specified in the <code>file.path</code> parameter into a local Docker registry.
<code>pushImage</code>	Pushes Integration Server image created for the on-premise Integration Server into the webMethods Cloud or Docker registry.
<code>help</code>	Displays usage information for each command.

Prerequisites for Building a Docker Image

Prior to building a Docker image for Integration Server, you must complete the following:

- Install Docker on the machine on which you are going to install Integration Server and start Docker as a daemon.
- Install Integration Server, packages, and fixes on a supported operating system using the instructions in *Installing Software AG Products*, and then configure Integration Server and the hosted products
- If you want to intend to run the Integration Server inside a Docker container in Integration Cloud and you want to expose some services to consumers, you must specify the Integration Server services before building the Docker image. For more information, see [“Specifying Services to Expose to Consumers in webMethods Cloud ” on page 955.](#)

Specifying Services to Expose to Consumers in webMethods Cloud

If you want to make Integration Server services running inside a Docker container in Integration Cloud available to consumers, you must complete additional configuration before creating the Docker image for Integration Server. Specifically, you must identify the Integration Server services that the Docker container exposes in Integration Cloud. Service invocation requests that originate from outside Integration Cloud cannot directly invoke services in the Docker container. Instead, the requests must go through the Integration Cloud endpoint which then redirects the request to the Docker container that exposes the requested service.

➤ To specify services to expose in Integration Cloud

1. Go to Integration Server Administrator and then go to **webMethods Cloud > Docker Services**.

The Package column lists all packages in Integration Server.

2. In the **Docker Services** column, specify the services to expose to consumers from your custom packages.

You can specify the service using one or more of the following.

- Directive and fully qualified name of the service in the format

/directive/folder.subfolder:serviceName

For example: `/invoke/is.assets:getChecksums`

- The URL for invoking the service.

For the URL, specify a snippet that is unique enough for Integration Cloud to accurately map it to the service in the Docker container.

- A URL alias created using **Settings > URL Aliases > Create Alias**

Use a comma to separate the services.

3. Click **Save Changes**.

Building the Docker Image for an Integration Server Instance

Building a Docker image for Integration Server consists of:

- Creating a Dockerfile for the Integration Server instance. This Dockerfile contains system (Wm*) packages and core functionality.
- Building a base Docker image for Integration Server using the Dockerfile with the core functionality and system packages.
- Creating a Dockerfile for the custom packages to include in the Integration Server image.

- Building a Docker image for Integration Server image using the custom packages Dockerfile. The resulting Docker image contains core functionality, system packages, and custom packages.

Note:

If you use automatic package deployment, you do not need to create an image for custom packages. Instead, create the base image and an image containing the Default package. Then use the automatic package deployment feature to add packages to the Microservices Runtime or Integration Server running in a Docker container. For more information about automatic package deployment, see *Developing Microservices with webMethods Microservices Runtime*.

Before building an Integration Server image to use with Docker containers, make sure to complete the prerequisites listed in [“Prerequisites for Building a Docker Image” on page 954](#).

For information, including prerequisites, about using an Integration Server running in a Docker container as the local development server for the local service development feature in , see *webMethods Service Development Help*.

Note:

The Dockerfile created by Integration Server contains metadata about the Integration Server instance used for the image in the Dockerfile. The metadata appears as a Label in the Dockerfile and includes the name, product version, product build number, and primary port of the instance.

➤ To create a Docker image for Integration Server

1. Go to the *Software AG_directory\IntegrationServer\docker* directory.
2. Create a Dockerfile for the core of an Integration Server instance using one of the commands below. The Dockerfile is created in *Software AG_directory*.
 - Create a Dockerfile that specifies your choice of JDK or JRE and as many Wm packages as you want to include (that is, system packages and hosted product packages, such as Trading Networks Server) by running the `createDockerFile` command below. The file is stored in the Software AG directory.

```
is_container.sh/bat createDockerfile [optional arguments]
```

Argument

`-Dimage.name= baseImageName`

Description

Optional. Base image such as ubuntu or centos:7.

Default: centos:7

For an Integration Server running on Windows, the `-Dimage.name` parameter should be set as follows:

```
-Dimage.name=mc.microsoft.com/windows/servercore:YourOsImageTag
```


Argument	Description
	<p>Where <i>YourOsImageTag</i> is the tag for the Windows image created for the OS version on which Integration Server runs.</p>
	<p>Note: Make sure you select a base image that is supported with any layered products installed on the Integration Server instance that you intend to include in the Docker image. For more information about supported base images, see <i>System Requirements for Software AG Products</i>.</p>
<p>-Dinstance.name= <i>instance_name</i></p>	<p>Optional. Integration Server instance to include in the image.</p> <p>Default: default (The instance named default.)</p>
<p>-Dport.list= <i>ports</i></p>	<p>Optional. Comma-separated list of the ports on the instance to expose in the image.</p> <p>Default: 5555,9999</p>
<p>-Dpackage.list= <i>Wm packages</i></p>	<p>Optional. Comma-separated list of Wm packages on the instance to include in the image.</p> <p>Default: all (this includes all of the Wm packages and the Default package)</p>
<p>-Dinclude.jdk={true false}</p>	<p>Optional. Whether to include the Integration Server JDK (true) or JRE (false) in the image.</p> <p>Default: true</p>
<p>-Dfile.name= <i>Dockerfile_name</i></p>	<p>Optional. Filename for the generated Dockerfile.</p> <p>Default: Dockerfile_IS</p>
<p>-Dtarget.configuration= <i>configuration_name</i></p>	<p>Optional. Target configuration of Integration Server for which the Dockerfile is being created.</p> <p>Default: There is no default value for this parameter.</p>

Argument	Description
	<p>If the Dockerfile is for an Integration Server that will be used as the local development server, specify: <code>localdev</code></p> <p>For more information about using an Integration Server running in a Docker container as the local development server for the local service development feature in <i>, see webMethods Service Development Help.</i></p> <p>If the Dockerfile is for an Integration Server that will be used with OpenShift, specify: <code>OpenShift</code></p> <p>When <code>-Dtarget.configuration=OpenShift</code>, the <code>is_container.bat/sh</code> script creates a root group that has read, write, and execute permissions. This is needed because OpenShift runs Docker containers using an arbitrarily assigned user ID. To support running of the container by an arbitrary user, a root group needs to own the directories and files that might be written to by the processes in the Docker image, the root group must have read/write access to the directories and files, and the root group must have execute permissions to the files that can be executed.</p> <div style="background-color: #f0f0f0; padding: 5px; border: 1px solid #ccc;"> <p>Note: The <code>OpenShift</code> option is available after applying a fix that includes PIE-60336 (IS_10.5_Core_Fix4 or later).</p> </div> <p>If the Dockerfile is for an Integration Server instance that will be lifted and shifted into Integration Cloud, set to: <code>wmcloud</code></p> <p>When this parameter is set to <code>wmcloud</code>, the <code>is_container</code> script overwrites the default truststore provided with Integration Server with the truststore required by Integration Cloud.</p>
<code>-Dimage.createUser={true false}</code>	<p>Optional. Sets a non-root user for the ownership of the files in the Docker image. When <code>-Dimage.createUser</code> is set to <code>true</code>, Integration Server creates a Docker image</p>

Argument	Description
	for which a sagadmin user has ownership of the files in the Docker image. The default value of the parameter is false.

Default: false

Note:

This option applies on operating systems other than Windows and requires Docker version 17.09 or higher.

- Create a Dockerfile that includes only a JRE and only the minimum system packages required for Integration Server to operate (WmRoot, WmPublic, and WmCloud) by running the `createLeanDockerfile` command below.

```
is_container.sh/bat createLeanDockerfile [optional arguments]
```

The optional arguments are `-Dimage.name`, `-Dinstance.name`, `-Dport.list`, `-Dfile.name`, `-Dtarget.configuration`, and `-Dimage.createUser`.

3. Build a Docker image with the core Dockerfile by running the `build` command below. The image is stored in the local Docker registry.

```
is_container.sh/bat build [optional arguments]
```

Argument	Description
<code>-Dfile.name= <i>Dockerfile_name</i></code>	Optional. Filename of the Dockerfile to use to build the Docker image. Default: Dockerfile_IS
<code>-Dimage.name= <i>Docker_image_name</i></code>	Optional. Name for the generated Docker image. Default: is:micro

4. Create a Dockerfile that specifies the custom packages to include in the Docker image by running the `createPackageDockerfile` command below. The Dockerfile for the custom packages is stored in the `Software AG_directory/Integration Server_directory/instances/instance_name/packages` directory.

```
is_container.sh/bat createPackageDockerfile [optional arguments]
```

Argument	Description
<code>-Dinstance.name= <i>instance_name</i></code>	Optional. Integration Server instance that includes the user-defined packages.

Argument	Description
	Default: default (The instance named default.)
<code>-Dimage.name= <i>Docker_image_name</i></code>	Optional. Name of the core Integration Server image on which this image should be built. Default: is:micro
<code>-Dpackage.list= <i>packages</i></code>	Optional. Comma-separated list of the custom packages to include in the image. Default: all “all” indicates that the image for packages will include all the custom packages on the Integration Server instance. The Default package and packages beginning with “Wm” will not be included in the image.
	Note: Make sure to include any custom packages containing services that you want to expose to consumers when Integration Server runs in a Docker container in Integration Cloud. If you do not include a package that contains an exposed service a request for the service fails with a service not found error when Integration Cloud redirects the request to the Docker container.
<code>-Dimage.createUser={true false}</code>	Optional. Sets a non-root user for the ownership of the files in the Docker image. When <code>-Dimage.createUser</code> is set to true, Integration Server creates a Docker image for which a sagadmin user has ownership of the files in the Docker image. The default value of the parameter is false.
	Note: This option applies on operating systems other than Windows and requires Docker version 17.09 or higher.
<code>-Dfile.name= <i>Dockerfile_name</i></code>	Optional. Filename for the Dockerfile that contains the custom packages. Default: Dockerfile_IS_Pkg

Note:

If you are using automatic package deployment you can skip the above step. Instead, once Microservices Runtime or Integration Server running in a Docker container, you can place custom packages in the folder on the HOST machine that is mapped to the autodeploy

folder. For more information, see *Developing Microservices with webMethods Microservices Runtime*.

- Build the Docker image with the custom packages Dockerfile by running the `buildPackage` command below. The image is stored in the local Docker registry. The resulting Docker image contains custom packages, system packages, and the core Integration Server functionality.

```
is_container.sh/bat buildPackage [optional arguments]
```

Argument	Description
<code>-Dfile.name= Dockerfile_name</code>	Optional. Filename for the Dockerfile that contains the custom packages. Default: <code>Dockerfile_IS_Pkg</code> Docker build uses the file with the specified name located in the packages directory of the specified instance, specifically: <i>Software AG_directory/Integration Server_directory /instances/instance name/packages/</i>
<code>-Dimage.name= Docker_image_name</code>	Optional. Name for the generated Docker image that contains the custom packages. Default: <code>is:microPkg</code>

- Optionally, based on your requirements, execute one or more of the following command line arguments:
 - `saveImage` to save an Integration Server image to a file. For more information about `saveImage`, see [“Loading a Docker Image to an On-Premise Docker Registry” on page 961](#).
 - `loadImage` to load an image to a Docker registry. For more information about `loadImage`, see [“Pushing a Docker Image to an On-Premise Docker Registry” on page 962](#)
 - `pushImage` to push an Integration Server image created for an on-premise Integration Server to the Integration Cloud or a Docker registry. For more information about pushing an image to a Docker registry or Integration Cloud, see [“Pushing a Docker Image to an On-Premise Docker Registry” on page 962](#) and [“Pushing the Docker Image to Integration Cloud” on page 969](#)

Loading a Docker Image to an On-Premise Docker Registry

To load a Docker image for Integration Server to an on-premise Docker registry, you first save the image as a tar file and then load the tar file into the registry.

For information about pushing a a Docker image to an on-premise Docker registry, see [“Pushing a Docker Image to an On-Premise Docker Registry” on page 962](#).

➤ To save and load a Docker image to an on-premise Docker registry

1. Save the Docker image as a tar file by running the `saveImage` command below. By default, the tar file is saved in the `Software AG_directory/ Integration Server_directory /docker/images` directory.

```
is_container.sh/bat saveImage -Dimage.name=input Docker image
```

```
[-Dfile.path=full path to output tar file]
```

2. Load the Docker image from the tar file into a Docker registry by running the `loadImage` command below.

```
is_container.sh/bat loadImage [-Dfile.path=full path to input tar file]
```

Pushing a Docker Image to an On-Premise Docker Registry

You can push the Docker image created for an on-premise Integration Server to an on-premise Docker registry using the `pushImage` command.

➤ To push a Docker image to an on-premise Docker registry

- Run the `pushImage` command below

```
is_container.sh/bat pushImage required arguments[optional argument]
```

Argument	Description
<code>-Duser= <i>userID</i></code>	User ID to access the registry.
<code>-Dpassword= <i>password</i></code>	Optional. Password to access the registry. If you do not supply the password using this argument, you will be prompted for it when the <code>pushImage</code> command executes.
<code>-Dserver= <i>registry_URL</i></code>	URL of the registry. Example: <code>docker.io</code>
<code>-Drepository.name= <i>repository_name</i></code>	Name of the repository into which to push the image.
<code>-Dimage.name= <i>Docker_image_name</i></code>	Name of the Docker image to push, such as the Integration Server image <code>is:microPkg</code> .

Running the Docker Image in an On-Premise Docker Container

Use the `docker run` command to run a Docker image, such as one created for Integration Server, in an on-premise Docker container. When the Docker image is for an Integration Server, running the Docker image starts up Integration Server.

When using the `docker run` command, you can explicitly map a port on the machine that hosts the Docker container to an exposed port in the Docker container. Alternatively, you can allow Docker to map the exposed port to any available port on the Docker host machine. For a Docker container for Integration Server, the exposed ports include the primary port and the diagnostic port. If you want to use Integration Server Administrator to connect to the Integration Server running in the Docker container, you must explicitly assign a host machine port to the Integration Server primary port for the container. If you allow Docker to associate the host machine port with the Integration Server port, you will not know which host machine port Docker associated with the Integration Server primary port. Similarly, if you want to be able to access Integration Server Administrator through the diagnostic port, you must explicitly map a port on the host machine to the diagnostic port exposed on the container.

Note:

Many container deployment solutions provide the ability to view the console logs STDOUT and STDERR for a container. When running a Docker image of Integration Server in a Docker container, Integration Server writes the server log to the console as well as to the `server.log` file.

» To run a Docker image in an on-premise Docker container

- Run the following command:

```
docker run -d --name Docker_container_name -p [host_primary_port:]primary_port
-p [host_diagnostic_port:]diagnostic_port -p [host:]other_exposed_port
environment_variables Docker_image
```

Where this placeholder	Specifies
<i>Docker_container_name</i>	Name of the Docker container in which you want to run the Docker image. For example, <code>IS_Default</code> .
<i>host_primary_port</i> :	Optional. A port on the container host machine that you want to explicitly map to an exposed port on the Docker container, specifically the primary port of Integration Server. If you do not specify <i>host_primary_port</i> , the Docker container maps any available port on the container host machine to the <i>primary_port</i> .
<i>primary_port</i>	Port number for the primary port on Integration Server.

Where this placeholder	Specifies
<i>host_diagnostic_port</i> :	Optional. A port on the container host machine that you want to explicitly map to an exposed port on the Docker container, specifically the diagnostic port of Integration Server. If you do not specify <i>host_diagnostic_port</i> , the Docker container maps any available port on the container host machine to the <i>diagnostic_port</i> .
<i>diagnostic_port</i>	Port number for the diagnostic port on Integration Server.
<i>host_port</i> :	Optional. A port on the container host machine that you want to explicitly map to an exposed port on the Docker container. If you do not specify <i>host_port</i> , the Docker container maps any available port on the container host machine to the <i>other_exposed_port</i> .
<i>other_exposed_port</i>	Port number for an exposed port on Integration Server.
<i>environment_variables</i>	Name=value pair for any environment variables that you want to use. For a list of Integration Server environment variables for use with Docker, see “Environment Variables for Use with Docker” on page 1183 .
<i>Docker_image</i>	Name of Docker image to run in the on-premise Docker container. For example, is:microPkg

Examples

Suppose your Docker image for an Integration Server exposes port 5555 and 9999.

In the following example, the command does not include ports on the container host machine. Consequently, the Docker container automatically maps the exposed ports to available ports on the container host machine.

```
docker run -d --name IS_Default -p 5555 -p 9999 is:microPkg
```

In the following example, the command explicitly maps ports available on the container host machine to exposed ports on the Docker container.

```
docker run -d --name IS_Default -p 34678:5555 -p 34679:9999 is:microPkg
```

Note:

You can find running instances of a Docker container by running the command: `docker ps -a`

Writing the Server Log to the Console

Many container deployment solutions provide the ability to view the console logs STDOUT and STDERR for a container. When running a Docker image of Integration Server in a Docker container, Integration Server writes the server log to the console as well as to the server.log file. Server log messages written to STDOUT include the identifier “ISSERVER” to help differentiate server log messages from other messages written to the console.

Docker includes logging drivers to help obtain information from running containers. If you use the json-file logging driver or journald logging driver with your Docker solution, the driver captures information written to STDOUT and writes it to a file. Running the `docker logs` command displays the contents of that file. The size of the file used for logging can grow. You may want to consider using logging driver options to limit the size of the file and prevent disk overflow issues.

For example, The size of the JSON file used for logging can grow indefinitely. That is, by default, there is no maximum limit to the size of this file. If your Docker solution uses the json-file logging driver, you may want to consider limiting the size of the JSON file by rotating the JSON log files.

To specify a size limit for the JSON log file or any other options for the json-file logging driver, supply one or more of the supported json-file logging options in the `docker run` command. For example, the following starts a Docker container named IS_1 using the `is:microPkg` image. The container can have a maximum of 4 log files with maximum size of 25MB each:

```
docker run -d --name IS_1 -p 55555:5555
--log-opt max-size=25m --log-opt max-file=4 is:microPkg
```

For information about the json-file logging driver or any other logging driver included with Docker, consult the Docker documentation.

Writing Audit Logs to the Console

When you start a Docker container for Integration Server, you can instruct the Integration Server to write some or all of the audit logs to the console (STDOUT). Writing audit log entries to the console provides streamlined logging which can be especially useful when Integration Server runs in a Docker container. When working with multiple containers, you can obtain the log entries generated by the containers from the console. This may be preferable to retrieving the log files from the file system inside each container.

Each audit log entry written to the console includes the following:

- The logger name as the identifier. For example, WMERROR is the identifier for the error log and WMSESSION is the identifier for the session log.
- The `||` (double pipe) character serves as the field delimiter. You can change the field delimiter using the `watt.server.audit.stdout.fieldDelimiter` server configuration parameter.
- The `&&` symbol serves as the record delimiter. You can change the record delimiter using the `watt.server.audit.stdout.recordDelimiter` server configuration parameter.

Following is an example of a session log entry written to the console:

```
jvm 1 |
&WMSESSION|81b74e40-910207b4|?|81b74e40-910207b4|91285d13-30d5a089|not.55|20662165-9967|20662165-9967|||Default|Internal
connection||0||0||8798bcf3e47849368c42677bd76bba91||?&&
```

To write some or all of the audit logs to the console when Integration Server runs in a Docker container, set the `SAG_IS_AUDIT_STDOUT_LOGGERS` environment variable when you start the Docker container with the `docker run` command.

Set the `SAG_IS_AUDIT_STDOUT_LOGGERS` to one of the following values:

- A comma-separated list of the audit loggers that you want to write to STDOUT. For example: `WMSESSION,WMERROR`

The audit logger name is the first portion of the audit log file name. For a list of audit log file names, see *webMethods Audit Logging Guide*.
- A comma-separated list of the logger names as they appear in the **Settings > Logging** page in Integration Server Administrator. For example: `Session Logger,Error Logger`
- `ALL` to write all log entries for all file-based audit loggers to STDOUT.
- `NONE` to indicate that none of the file-based audit loggers will write to STDOUT. This is the default value.

Note:

The value of `SAG_IS_AUDIT_STDOUT_LOGGERS` is not case-sensitive.

Integration Server considers writing audit log entries to STDOUT to be an auxiliary output which means the logger still writes to the specified audit log destination of file or database.

Accessing an Integration Server Running in a Docker Container

When Integration Server runs in a Docker container, you can access the Integration Server using Integration Server Administrator.

To use Integration Server Administrator with an Integration Server running in the Docker container, open an Internet browser and point the browser to the Integration Server port on the container host machine on which Integration Server runs. For more information, refer to the Docker documentation.

Note:

You can use Integration Server Administrator with an Integration Server running in the Docker container only if you explicitly mapped a port on the machine that hosts the Docker container to an exposed port in the Docker container when executing the `docker run` command. That is, a port on the container host machine must be mapped to a port on Integration Server. For more information about running the Docker image for Integration Server in a Docker container, see [“Running the Docker Image in an On-Premise Docker Container”](#) on page 963.

➤ **To start the Integration Server Administrator for an Integration Server running in a Docker container**

1. Start your browser.
2. Point your browser to the container host machine and Integration Server port exposed on the container using the following format:

protocol://containerHostMachine:host_port

For example, if you explicitly mapped the container host machine port 34678 to the primary port 5555 on Integration Server and the container host machine is named machineABC, you would type: `http:// machineABC: 34678`

3. Log on to the Integration Server with a username and password that has administrator privileges.

Externalizing Log and Configuration Files When Running Integration Server in a Docker Container

When running Integration Server in a Docker container, Integration Server supports the use of Docker volumes to persist configuration and log files to a mounted directory on the host file system. A Docker volume exists outside of the Docker container on the file system of the host machine, making it accessible directly. Docker volumes can be used for externalizing the contents of the logs directory for the Integration Server instance and the Integration Server profile as well as the config directory for the Integration Server instance running in the Docker container.

Note:

Many container deployment solutions provide the ability to view the console logs STDOUT and STDERR for a container. When running a Docker image of Integration Server in a Docker container, Integration Server writes the server log to the console as well as to the server.log file. You can also use the `SAG_IS_AUDIT_STDOUT_LOGGERS` environment variable to indicate which file-based audit logs will write to STDOUT. For more information, see [“Writing Audit Logs to the Console” on page 965](#).

To make use of volumes for configuration files and log files, supply one or more of the following environment variables when executing the `docker run` command. Include the environment variables in the `docker run` command immediately before the image name.

Environment Variable	Description
<code>HOST_DIR</code>	The path to the mounted directory on the HOST machine to which to write the files. Providing a <code>HOST_DIR</code> parameter and value with the <code>docker run</code> command externalizes the logs and configuration artifacts of Integration Server.
<code>SERVICE_NAME</code>	Name of a unique directory under <code>HOST_DIR</code> for persisting Integration Server artifacts. When supplied, the logs and config directories are created under the <code><HOST_DIR>/<SERVICE_NAME></code> directory and all artifacts

Environment Variable	Description
PERSIST_LOGS	<p>of logs and config directories are persisted in the respective directories. If <code>SERVICE_NAME</code> is not supplied, Integration Server instance name will be used for the name of the unique directory that gets created under the <code><HOST_DIR></code> directory. Software AG recommends specifying a unique directory for each Docker container.</p> <p>If set to true, the Integration Server running inside the Docker container persists the log files to <code><HOST_DIR>/<SERVICE_NAME>/logs</code> where <code><SERVICE_NAME></code> is replaced by <code><INSTANCE_NAME></code> if the <code>SERVICE_NAME</code> environment variable is not specified. Integration Server externalizes the logs located in the <code>Integration Server_directory/instances/<instanceName>/logs</code> and <code>profiles/IS_<instanceName>/logs</code> directories. If set to false, the Integration Server running inside the Docker container does not externalize the log files. The default is true.</p>
PERSIST_CONFIGS	<p>If set to true, the Integration Server running inside the Docker container persists the configuration files to the <code><HOST_DIR>/<SERVICE_NAME>/config</code> directory where <code><SERVICE_NAME></code> is replaced by <code><INSTANCE_NAME></code> if the <code>SERVICE_NAME</code> environment variable is not specified. If set to false, the Integration Server running inside the Docker container does not externalize the config files. The default is true.</p>

Example

Consider the following docker run command:

```
docker run -d --name IS_Default -p 5555 -p 9999
-v /opt/myfolder:/opt/myfolder
-e HOST_DIR=/opt/myfolder -e SERVICE_NAME=demo is:microPkg
```

The configuration and log files will be written to the directory `/opt/myfolder/demo` with log files written to `/opt/myfolder/demo/logs` and configuration files written to `/opt/myfolder/demo/config`. Note that the `opt/myfolder/demo/logs` directory contains the log files from the `Integration Server_directory/instances/instanceName/logs` directory and the `profiles/IS_default/logs` directory.

Note:

In the example above, you need to supply `-v /opt/myfolder:/opt/myfolder` only if the folder is not mounted on the machine executing the Docker container and is not visible to the Docker container.

Pushing the Docker Image to Integration Cloud

Use the `pushImage` command to push the Docker image created for an on-premise Integration Server to a Docker registry that is hosted by Integration Cloud.

Note:

Before you push a Docker image for Integration Server to webMethods Cloud, a repository for the Integration Server image must exist on webMethods Cloud for the stage to which you want to push the image. Furthermore, the repository name must correspond to the image name portion of the Docker image name. For example, in the Docker image name “is:microPkg”, “is” is the image name and “microPkg” is the tag given to the image name. Before you push the Docker image to Integration Cloud, a repository named “is” must exist on Integration Cloud for the specified stage.

➤ To push the Docker image to Integration Cloud

1. Navigate to the following directory:

```
Software AG_directory/ Integration Server_directory /docker
```

2. Run the following command:

```
is_container.sh/bat pushImage required arguments [optional arguments]
```

Argument	Description
-Duser= <i>user ID</i>	Username to access Integration Cloud.
-Dpassword= <i>password</i>	Optional. Password to access Integration Cloud. You will be prompted to provide a password later if you do not provide one with the <code>pushImage</code> command.
-Dserver= <i>URL of Integration Cloud</i>	URL for Integration Cloud. Example: <i>subDomain name.webmethodscloud.com</i>
-Dsubdomain.name= <i>subdomain Name</i>	Subdomain for Integration Cloud. Example: <i>subdomain name.webmethodscloud.com</i>
-Dstage.name={ <i>development test prelive live</i> }	Stage to which to push the image.
-Dimage.name= <i>Docker_image_name</i>	Name of the Docker image to push, such as the Integration Server image. Example: <i>is:microPkg</i>

Running the Docker Image in Integration Cloud

For information and instructions about managing Docker containers and repositories in Integration Cloud, see the *webMethods Integration Cloud Help* on working with Docker containers.

Stopping a Docker Container for Integration Server

The `docker stop` command stops a running container. By default, the `docker stop` command waits 10 seconds for a container to stop before Docker kills the container. However, the default wait time of 10 seconds is not sufficient for Integration Server to complete a graceful shut down. When using the `docker stop` command, use the `--time` option (`-t`) to specify a longer wait time that will allow Integration Server to shut down gracefully.

The number of seconds an Integration Server needs to complete a graceful shut down depends on the number of shutdown services configured for Integration Server and the amount of time it takes for those shutdown services to execute to completion. Software AG recommends that at a minimum, for an Integration Server with no shutdown services, set a wait time to at least 30 seconds. Increase the wait time based on the number of shut down services and the time required for those services to execute.

Using a Configuration Variables Template with a Docker Image

Microservices Runtime or an Integration Server equipped with an Microservices Runtime license provides the ability to use a configuration variables template to set configuration information for a Microservices Runtime running in a Docker container. A configuration variables template is a properties file that contains configuration data as a series of key-value pairs where the key name reflects the asset and particular asset property for which you can supply a value. The property values are set in the template and then applied by a Microservices Runtime at start up.

By externalizing configuration information, a single Docker image created for a Microservices Runtime or Integration Server can be used across multiple environments, including different stages of the production cycle. For example, you might use different templates for specific environments such as testing versus production. Or you might use the same template for all environments but use environment variables to vary the configuration in each environment.

For more information about configuration variable templates, see *Developing Microservices with webMethods Microservices Runtime*.

56 Integration Server Administrator API

■ Introduction to the Administrator API	972
■ Authentication and Authorization	972
■ REST URL Structure	972
■ CRUD Operations	973
■ Administrative Actions	974
■ Credentials	974
■ Expanding Responses	974
■ Media Type for Responses	976
■ Media Type for Request Payloads	976
■ Getting Started	976
■ Controlling Access to the Administrator API	977
■ Logging Client Errors	977
■ Including a Stack Trace in the Response Body	978
■ Integration Server Administrator API Operations	978

Introduction to the Administrator API

Integration Server provides an administrative API that can be used to perform administrative actions such as restarting Integration Server as well as creating, retrieving, updating and deleting assets on Integration Server. The Administrator API uses the REST architectural style. The unit of work for all REST operations is a resource. In the Integration Server Administrator API, resources are administrative assets such as packages, license keys, and the server itself.

The Integration Server Administrator API uses JSON as the media type. However, The API supports other media types requested by the client as long as Integration Server has a content handler for the media type.

All Administrator API requests and responses are described by an OpenAPI 3.0 document. This document can be obtained in the following ways:

- Requested from the API with `GET /admin/swagger/integrationServer`
- From the file system: `Integration Server_directory`
`\instances\instanceName\packages\WmAdmin\resources\wmadmin_swagger.json`

Note:

The OpenAPI Specification was originally called the Swagger specification. An OpenAPI document may also be referred to as a Swagger document or file or simply Swagger.

The Administrator API is available when the WmAdmin package is enabled. This package is enabled by default. Disabling the WmAdmin package disables the API.

Using the Integration Server Administrator API requires familiarity with REST, OpenAPI 3.0, and Integration Server.

Authentication and Authorization

The Administrator API supports any form of authentication and authorization supported by Integration Server.

REST URL Structure

The Administrator API makes resources accessible through a URL path. All requests to the Administrator API must be issued using HTTPS or HTTP. The Administrator API supports the standard HTTP methods GET, POST, PUT, PATCH, and DELETE. Not all resources will support all HTTP methods. However, all resources support the HEAD and OPTIONS methods.

An absolute URL for an Administrator API resource has the following structure:

```
https://host:port/admin/resourceType[/resourceId]
```

Where:

- `https` is the transport protocol. You can use HTTP or HTTPS.

Software AG recommends using the more secure HTTPS protocol to administer your Integration Server.

- *host:port* is the host and port of the Integration Server you want to administer.
- *admin* is the directive for the Administrator API.
- *resourceType* is the type of resource, such as package.
- *resourceId* identifies a specific resource. *resourceId* is only used with some requests.

For example:

```
HTTP GET http://localhost:port/admin/package
```

Retrieves all packages

```
HTTP GET http://localhost:port/admin/package/WmPublic
```

Retrieves the WmPublic package

The relative URL for an Administrator API resource must start with `/admin` and then identify the type of resource. The relative URI has the following structure:

```
/admin/resourceType[/resourceId]
```

For example:

```
GET /admin/license
```

Returns license information for Integration Server.

CRUD Operations

CRUD (Create Retrieve Update Delete) operations are one type of request supported by the Administrator REST API. CRUD operations are executed using different HTTP methods.

The following table identifies the HTTP method and the related CRUD operation.

Method	Operation
GET	Retrieves one or more resources
POST	Creates a new resource with the details sent in the request.
PUT	Replaces the resource with the details sent in the request.
PATCH	Updates a resource by replacing the fields provided in the body of the request.
DELETE	Deletes a resource.

Note:

Not all HTTP methods are available for every resource. For information about which methods are available for a resource, review the API descriptions in the OpenAPI 3.0 document for the Administrator API. All resources support the HEAD and OPTIONS methods.

Administrative Actions

Administrative actions such as restarting the server and enabling a package are another type of request supported by the Administrator API. Requests for administrative actions use the POST method. The URL path identifies the resource on which to act. The URL query has an action parameter that identifies the action to take.

For example:

```
POST /admin/package/WarehouseUtils?action=enable
```

Enables the WarehouseUtils package.

```
POST /admin/server?action=restart
```

Restarts Integration Server.

Administrative actions may have additional query parameters.

For example:

```
POST /admin/server?action=shutdown&force=true
```

Shuts down Integration Server immediately, even if there are active sessions.

Credentials

The Administrator API is stateless. A session is not saved on Integration Server when a request is processed. A Set-Cookie header is not returned in the response. Administrator clients must provide credentials with each request.

```
GET /admin/license HTTP/1.1
Host: host:port
Authorization: Basic <base64-encoded username:password>
Accept: application/json
```

Retrieves license information.

Expanding Responses

Some Administrator API responses contain multiple resources. For example, for the request GET /admin/package, Integration Server returns all packages. By default, the response will consist of an array of links

```
{
  "packages": [
    {
      "name": "Default",
      "url": "https://localhost:5543/admin/package/Default/"
    }
  ]
}
```

```

    },
    {
      "name": "WmAdmin",
      "url": "https://localhost:5543/admin/package/WmAdmin/"
    },
    {
      "name": "WmART",
      "url": "https://localhost:5543/admin/package/WmART/"
    },
    {
      "name": "WmAssetPublisher",
      "url": "https://localhost:5543/admin/package/WmAssetPublisher/"
    },
    {
      "name": "WmCloud",
      "url": "https://localhost:5543/admin/package/WmCloud/"
    },
    },
    etc...
  ]
}

```

Rather than getting an array of links, you can request that the Administrator API return the complete package resources by including the query parameter `expand=true` in the URL.

```
GET /admin/package?expand=true
```

In this case, the response will look something like this:

```

{
  "packages": [
    {
      "packageName": "Default",
      "loadok": "1",
      "loaderr": "0",
      "loadwarning": "0",
      "version": "10.5.0.0.5015",
      "build": "5015",
      "description": "",
      "patch_nums": "",
      "jvm_version": "1.8",
      "publisher": "Software AG",
      "time": "",
      "listACL": "Default",
      "message": "",
      "code": "1"
    },
    {
      "packageName": "WmAdmin",
      "loadok": "2",
      "loaderr": "0",
      "loadwarning": "0",
      "startupServices": [
        "wm.admin.controller:init"
      ],
      "version": "10.5.0.0.5015",
      "build": "5015",

```

```
"description": "Administrative API",  
"patch_nums": null,
```

```
"jvm_version": "1.8",  
"publisher": "Software AG",  
"time": null,  
"listACL": "Administrators",  
"successors": [  
  "WmRoot;10.5.0"
```

```
],  
"message": "",  
"code": "1"  
},  
etc...  
]
```

```
}
```

Media Type for Responses

API clients can control the media type of the response by including an Accept header in the request.

The following request retrieves license information from Integration Server

```
GET /admin/license HTTP/1.1  
Host: host:port  
Authorization: Basic <base64-encoded username:password>  
Accept: application/json
```

Media Type for Request Payloads

Requests that include a payload, such as PUT, PATCH and POST, must include a Content-type header to indicate the media type of the request body.

The following request updates the Integration Server license key file.

```
PATCH /admin/license HTTP/1.1  
Host: host:port  
Authorization: Basic <base64-encoded username:password>  
Content-type: application/json  
Accept: application/json  
  
{  
  "licenseKeyFile" : "/user/sysAdmin/licenseKey.xml"  
}
```

Getting Started

To get started using the Administrator API, you must do the following:

1. Make sure the Administrator API is enabled by verifying that the WmAdmin package is enabled.

2. Configure access to the Administrator API by assigning administrator privileges to users who can perform CRUD operations and read-only permissions to users who can only retrieve information. For more information, see [“Controlling Access to the Administrator API” on page 977](#).
3. Configure logging and error handling by determining whether Administrator API client errors are written to the error log and if the stack trace is included in error response to the client. For more information, see [“Controlling Access to the Administrator API” on page 977](#) and [“Including a Stack Trace in the Response Body” on page 978](#).

Controlling Access to the Administrator API

The Administrator API provides two levels of permissions:

- Administrators, that is members of the Administrators group have full access to the Administrator API. Users who belong to the Administrators group may perform all CRUD operations and administrative actions.
- Read-only administrators who may only issue GET requests.

If a read-only administrator uses any other HTTP method in the Administrator API, the server rejects the request with a 403 status code. Read-only administrators belong to a user group that is assigned to the `watt.adminapi.group.readOnly` server configuration parameter.

Note:

If a client who is not an administrator or a read-only administrator issues an Administrator API request, the server rejects the request with a 403 status code.

Creating a Read-Only Administrators Group

A read-only administrators group can only access the Integration Server Administrator API using GET requests.

➤ To create a read-only administrators group

1. Create a user group. Add users that will be read-only administrators to the group. For more information about creating user groups, see [“Defining Groups” on page 98](#).
2. Use extended settings to set the value of the `watt.adminapi.group.readOnly` to the name of the user group. User group names are case-sensitive.
3. Reload the WmAdmin package for changes to take effect. Alternatively, restart Integration Server.

Logging Client Errors

Integration Server always writes server errors that occur during execution of REST resources in the Administrator API to the error log. A server error is one that results in an HTTP status code

in the 500s. However, client errors, those that result in an HTTP status code in the 400s, are not written to the error log by default. If you want these errors to be written to the error log, set `watt.adminapi.log.clientErrors` true. Changes to this property take effect immediately.

Including a Stack Trace in the Response Body

When the Administrator API encounters an exception, the error is returned to the client in an HTTP response. By default, the body of the response contains the exception message and the stack trace which can be useful for debugging purposes. If you want error responses to omit the stack trace, set `watt.adminapi.returnExceptions` to false. Set `watt.adminapi.returnExceptions` to true, the default, to include the stack trace in the response body. Changes to this property take effect immediately.

Integration Server Administrator API Operations

The Integration Server Administrator API contains the operations listed in the following table.

Operation	Description
GET /admin/license/	Retrieves license information.
POST /admin/license/	Updates Terracotta license file or Integration Server license key file.
PUT /admin/license/	Updates Terracotta license file or Integration Server license key file.
PATCH /admin/license/	Updates Terracotta license file or Integration Server license key file.
GET /admin/package	Retrieves all packages.
GET /admin/package/{packageName}	Gets package information for a particular package.
POST /admin/package/{packageName}	Performs an administrative action on an installed package where the action can be: disable, enable, reload, or activate.
GET /admin/server	Retrieves information about Integration Server.
POST /admin/server	Performs administrative actions on Integration Server where the action can be: stop, restart, restartQuiesce, quiesce, exitQuiesce.
GET /admin/server/diagnostics	Returns a diagnostic archive as a ZIP attachment.
GET /admin/server/updates	Retrieves information about fixes installed on Integration Server.
GET /admin/swagger/{productName}	Retrieves the Swagger document for a product's API.

Operation

GET /admin/swagger

Description

Retrieves the Swagger documents for all administrative APIs in Integration Server.

If Integration Server is the only installed product with an Administrator API, the Integration Server's Swagger document is returned (same as GET /admin/swagger/integrationServer).

57 Simulating Metering in Integration Server

■ About webMethods Metering	982
■ About the Metering Simulator	983
■ Enabling the Metering Simulator	984

Integration Server includes a metering simulator that, when enabled, will generate server log messages that let you see the transaction usage that will be sent to the metering server if all the metering criteria is met.

Note:

The metering simulator was introduced with PIE-67717 (IS_10.5_Core_Fix9).

About webMethods Metering

Software AG provides a subscription, usage-based pricing model for many of the company's products, including Integration Server. The webMethods Metering utility uses a local webMethods Metering Agent installed with a product to collect data based on the webMethods product usage and then sends the data to the webMethods Metering Server which handles the billing aspect of the product usage. The criteria for transaction metrics to be reported to webMethods Metering include being on one of the supported versions, applying the most recent fixes, and having a license key installed that is associated with a transaction-based contract.

Integration Server regularly checks the status of the local webMethods Metering Agent. During start up, Integration Server checks the license to see if it is transaction-based. If so, Integration Server gathers status about the following:

- The local webMethods Metering Agent configuration in *Integration Server_directory* \common\metering\conf\metering.agent.properties
- The webMethods Metering Agent log files contained in *Integration Server_directory* \common\metering\storage. Integration Server verifies the contents of these log files and checks the format of the entries. The webMethods Metering Agent log files contain the transaction information that is sent to the webMethods Metering Server.
- The webMethods Metering Agent version.

If any of the above status checks fail, Integration Server writes an error message to the error log and server log. Integration Server Administrator displays a notification about the failed status check. Integration Server writes successful status check messages to the server log when the 0176 Metering server log facility is set to Trace.

After completing the local webMethods Metering Agent status checks during start up, Integration Server performs a "dry run" of sending data to the webMethods Metering Server. During a dry run, Integration Server simulates a connection to the webMethods Metering Server. If the dry run fails, Integration Server writes error message to the error log and server log. Integration Server Administrator displays a notification about the failed dry run. If the dry run succeeds, Integration Server writes success messages to the server log when the 0176 Metering server log facility is set to Trace.

After the initial dry run executes during start up, Integration Server executes the system task Metering Agent Diagnostic Log at the interval determined by the `watt.server.meteringAgent.logInterval`. The default interval between dry runs is 60 minutes.

For more information on webMethods Metering, consult the *Software AG Infrastructure Administrator's Guide*.

Note:

The metering agent “dry run” behavior was introduced with PIE-69024 (IS_10.5_Core_Fix9).

About the Metering Simulator

Integration Server includes a metering simulator that gathers service metrics and associated transaction information and writes that information to the server log. When the metering simulator is enabled, Integration Server produces a server log message for each service that would be metered. Integration Server writes one log message per invocation of a top-level, user-defined service. Integration Server does not aggregate data across multiple invocations of the same service into a single log message.

The metering simulator (and the webMethods Metering Agent) generates metrics for top-level user-defined services only.

Following is an example of server log message for a metered service:

```
2021-05-20 16:55:35 EDT [ISS.0176.9999I] (tid=92) Simulated metering metrics for service:
myFolder.mySubFolder:exampleService, tenant=null, cpu=0(ns), duration=0(ms),
transactions=1
```

The format for the message is:

```
Simulated metering metrics for service:serviceName, tenant=ID, cpu=cputime(ns),
duration=durtime(ms), transactions=count
```

Where:

- *serviceName*: The name of the top-level service for which the simulator is generating metrics.
- *tenant*: The tenant ID. For a self-hosted runtime, the tenant ID will be null.
- *cpu*: The amount of CPU time used for the service, measured in nanoseconds.
- *duration*: The duration of the service, measured in milliseconds.
- *transactions*: The number of transactions counted for this single service invocation. If the duration is less than 3 seconds, the count will be 1. If the duration exceeds 3 seconds, additional transactions will be counted for each 3 second interval. If your license key defines a different transaction duration value, the simulator will use that value to determine the number of transactions.

Note:

The (tid) is the thread ID of the thread performing the service execution.

The metering simulator runs independently of webMethods Metering. The metering simulator only enables the logging of messages for services that would be metered and therefore generate transaction data. Enabling the metering simulator does NOT result in actual reporting of the measured transaction data nor does it result in charges.

The metering simulator has the following main use cases:

- Your site has the traditional perpetual license and you are considering moving to the transaction-based pricing model. Enabling the metering simulator for a period of time, and then analyzing the resulting data may provide guidance on how many transactions per month your site would need. The period of time for which you enable the simulator should reflect a typical transaction load.
- Your site already has a subscription, usage-based license but you want access to critical metering data that is not reported with webMethods Metering. Specifically, the metering simulator provides the names of the top-level services that generate the transactions and the number of transactions resulting from each top-level service execution. The logs generated by metering simulator can be key to understanding your usage data. For example, if you see a webMethods Metering report that indicates your site ran a million transactions last month, you may want to know the source of those transactions: which Integration Server generated the most transactions and which services are long-running and contributing the most transactions. Using the server logs created while the metering simulator is running can help you determine the answer to these questions.

Enabling the Metering Simulator

Enabling and disabling of the metering simulator is controlled by the `watt.server.metering.simulator.enabled` server configuration parameter. Set this parameter to true to have Integration Server gather metering data for top-level services and write metering simulation messages to the server log. Set to false to disable the metering simulator. The default is false. You do not need to restart Integration Server for changes to this parameter to take effect.

58 Using Command Central to Manage Integration Server

■ An Overview of Command Central	986
■ Integration Server Instance Management	986
■ Accessing Integration Server Administrator through Command Central	992
■ Monitoring Integration Server Instances	994
■ Integration Server Configuration Types	996
■ Lifecycle Actions for Integration Server	1003
■ Commands that Integration Server Supports	1003
■ Using Unix Shell Scripts to Change Connection Credentials for Managed Products ..	1004
■ Command Central Command Line Tool Upgrade	1005

This chapter explains how to create, manage, and monitor Integration Server instances in Command Central. For details on the Integration Server configuration types supported in Command Central, see the documentation for the corresponding feature in the Integration Server Administrator's Guide this guide.

For example, the SFTP Server Alias configuration type in Integration Server, which is supported in Command Central is documented in [“Creating an SFTP Server Alias” on page 446](#). Similarly, the SFTP User Alias configuration type is documented in [“Creating an SFTP User Alias” on page 450](#).

Note:

Command Central uses Software AG Platform Manager to communicate with Integration Server. Software AG Platform Manager maintains an active connection with Integration Server and pings the Integration Server instance at a regular interval of 60 seconds.

An Overview of Command Central

Command Central is a centralized management tool that simplifies product installation, fix management, configuration across instances, and deployment of assets across Software AG products.

Using Command Central, you can manage Integration Server installations across environments, create, configure, and monitor multiple instances from one place.

Integration Server Instance Management

You can manage Integration Server instances using Command Central instance management commands and Command Central web user interface.

Note: Command Central cannot be used to manage Microservices Runtime.

Creating Instances Using Command Central Web User Interface

You can create an Integration Server instance using the Command Central web user interface:



➤ **To create an instance**

1. In the **Environments** pane, select the environment in which you want to create the new product instance .
2. Click the **Installations** tab.
3. Select the installation in which you want to create the new instance.
4. Click **Instances** tab.

5. Click  and select Integration Server.

6. Provide the following information and click **Next**.

In this field	Specify
Instance name	The name of the new instance.
IP Address	Optional. Default IP address to which to bind ports on this instance of Integration Server. Specify a bind address if your machine has multiple IP addresses and you want to use the same port number across different instances of Integration Server on the machine. In this case, each instance must specify a different bind address. If you do not specify a default bind address, the port listens to all network interfaces which prevents the same port number from being used on multiple instances.
	Note: Integration Server uses the supplied <code>default_bind_address</code> value as the value of the <code>watt.server.inetaddress</code> configuration property.
License key file	Optional. The location of your Integration Server license file.
Register Windows service for automatic startup	Optional. Whether to register the service for the instance. The default is <code>false</code> .
Database	<p>Database type The type of database used by the new server instance. Specify one of the following databases:</p> <ul style="list-style-type: none"> ■ <code>sqlserver</code> - Microsoft SQL Server ■ <code>oracle</code> - Oracle ■ <code>db2</code> - DB2 <p>These values are not case-sensitive. The default is the embedded IS internal database.</p> <p>JDBC URL The connection URL for the database.</p> <p>Database user The user name assigned to the Integration Server database.</p> <p>Password The password of the Integration Server database user.</p> <p>Connection name The connection name (alias name) for the database.</p>


In this field	Specify
Ports	<p>Primary port Optional.</p> <p>A default HTTP port number for the new instance. The port number must be unique for each Integration Server instance. The default value is 5555.</p>
	<p>Diagnostic port Optional. A default diagnostic port number for the new instance. The port number must be unique for each Integration Server instance. The default value is 9999.</p>
	<p>JMX port Optional. A default JMX port number for the new instance. The port number must be unique for each Integration Server instance. The default value is 8075.</p>
Packages to deploy to this instance	<p>The additional packages to add to the server instance.</p> <p>Select the package from the Available pane and click  to move the packages to the Selected pane. To add all the packages listed in the Available pane, click .</p>



7. Click **Finish**.

Updating Instances Using Command Central Web User Interface

You can update an Integration Server instance using the Command Central web user interface:

> To update an instance

1. In the **Environments** pane, select the environment in which you want to create the new product instance .
2. Click the **Installations** tab.
3. Select the installation which contains the instance.
4. Click **Instances** tab.
5. Select the Integration Server profile and click  .

6. Select **Update Instance**.
7. Perform the following and click **Next**.
 - To add the additional packages, click **Packages to add to this instance** and select the package from the **Available** pane and click  to move the packages to the **Selected** pane. To add all the packages listed in the **Available** pane, click . select the additional packages that you want to add.

Note:

Packages to add to this instance dialog box does not list the packages that are already installed in the instance.

- To update the database properties, click **Database** tab and enter the details.

Note:


For more information on Database properties, see [“Creating Instances Using Command Central Web User Interface”](#) on page 986.

8. Click **Finish**.

Deleting Instances Using Command Central Web User Interface

You can delete an Integration Server instance using the Command Central web user interface:

> To delete an instance

1. In the **Environments** pane, select the environment in which you want to create the new product instance .
2. Click the **Installations** tab.
3. Select the installation which contains the instance to be deleted.
4. Click **Instances** tab.
5. From the Integration Server profile, select the Integration Server instance and click .
6. Click **OK**.

Managing Instances Using the Command Central Instance Management Commands

The following table lists the required parameters that you must include when managing Integration Server instances using the Command Central instance management commands:

Command	Parameter	Description
sagcc create instances	instance.name= <i>name</i>	Required. A name for the new Integration Server instance.
	primary.port= <i>port</i>	Required. The main listening port for the new Integration Server instance.
	diagnostic.port= <i>port</i>	Required. The diagnostic port for the new Integration Server instance.
	jmx.port= <i>port</i>	Optional. The JMX port for the new Integration Server instance.
	install.service= false true	Optional. Specifies whether to install Integration Server as an application or a service. Valid values: <ul style="list-style-type: none"> ■ true - install as a service ■ false - install as an application (default)
	license.file= <i>location_of_license_file</i>	Optional. The location of your Integration Server license file. The script copies the license file from the specified location to <code>IntegrationServerInstanceName\bin\license.xml</code> file. This parameter is optional; however, if a license file location is not specified, the Integration Server instance will shut down after 30 minutes. Note: If the location name contains spaces, you must surround the name of the location with quotation marks (" ").
	db.alias= <i>database_alias_name</i>	Optional. The alias name for the database. The default value is JDBC_POOL_ALIAS.

Command	Parameter	Description
	<code>db.type=database_type</code>	<p>Required. The type of database used by the new server instance. Specify one of the following databases:</p> <ul style="list-style-type: none"> ■ <code>sqlserver</code> - Microsoft SQL Server ■ <code>oracle</code> - Oracle ■ <code>db2</code> - DB2 <p>These values are not case-sensitive. The default is the embedded IS internal database.</p>
	<code>db.username=database_username</code>	Required. The user name assigned to the Integration Server database.
	<code>db.password=database_password</code>	Required. The password of the Integration Server database.
	<code>db.url=database_URL</code>	Required. The connection URL for the database.

Example When Executing on Command Central

- To create the new instance for an installed Integration Server with instance name “is-instance2”, diagnostic port “8083”, and primary port “8081” in the installation with alias name “productionNode2”:

```
sagcc create instances productionNode2 integrationServer
instance.name=is-instance2 diagnostic.port=8083
primary.port=8081
db.alias=sql_server db.type=sqlserver
db.username=sa db.password=password@1234
db.url="jdbc:wm:sqlserver://10.60.72.87:1433;databaseName=SQL_DB"
```

Examples When Executing on Platform Manager

- To create the new instance for an installed Integration Server with instance name “is-instance2”, diagnostic port “8083”, JMX port “10058”, and primary port “8081”:

```
sagcc create instances integrationServer instance.name=is-instance2
diagnostic.port=8083 jmx.port=10058 primary.port=8081
db.alias=sql_server db.type=sqlserver
db.username=sa db.password=password@1234
db.url="jdbc:wm:sqlserver://10.60.72.87:1433;databaseName=SQL_DB"
```

- To create the new instance for an installed Integration Server using the instance data in the instance-settings.properties file, located it in the current directory:

```
sagcc create instances integrationServer -i instance-settings.properties
```

- To create the new instance for an installed Integration Server using the instance data in the `instance.settings.xml` file, located in the current directory:

```
sagcc create instances integrationServer -i instance-settings.xml
```

For more information about Software AG Command Central, see Command Central documentation.

Accessing Integration Server Administrator through Command Central

In Command Central, single sign-on (SSO) is designed to manage webMethods products using an administrative link without any post-installation configuration. When performing advanced configuration tasks, you might need to access the product's primary administrative interface. Command Central provides a link to the administrative interface on the Instances Overview page for each managed product. For example, when you click the Integration Server link on the Overview page of an Integration Server instance, Command Central redirects the browser to the corresponding Integration Server Administrator URL.

You must have administrative credentials to access the Command Central web user interface's administration links to Integration Server.

Important:

Use *only* one Command Central instance to manage a landscape. You cannot access the Command Central web user interface for a Command Central instance from another Command Central instance.

For information about generating and configuring custom SSO and SAML certificates for Software AG Common Platform-based products, see the Software AG Security Infrastructure documentation.

Changing the Authentication Mode

By default, the Authentication mode for run-time components that support trusted authentication is set to **Trusted**.

In the instance **Overview** tab, click  in the **Authentication** field to change the authentication mode using the Authentication Mode dialog box.

When you specify the authentication mode for an Integration Server instance, that authentication mode is also set for all layered product instances of the main product instance. However, changing the authentication mode for the OSGI profile of Integration Server does not change the authentication mode for the Integration Server run-time component that belongs to that OSGI profile.


To use basic authentication, you must change the authentication mode for a run-time component to **Fixed User**. Command Central uses basic authentication with a fixed user to communicate with Platform Manager. With **Fixed User** authentication, the authentication credentials for the Platform Manager will be fixed.

Changing the Administrator User Password for Integration Server in Command Central

You change the Administrator password for an Integration Server managed by Command Central in the Command Central web user interface. After changing the Administrator password for a managed product in Command Central, the outbound credentials are updated automatically.

➤ To change the Administrator user password for a product in Command Central

1. In the Environments pane in Command Central, select the environment that contains the managed product instance.
2. In the Instances table, select the Integration Server component under `integrationServer-instancename`, for example `integrationServer-default`
3. On the Configuration tab, select **Users**.
4. On the Users page, click **Administrator**.
5. Click **Edit** and specify the new password for **Administrator**.

To verify the new credentials click  and check the monitoring KPIs for the product component.

Configuring Integration Server for SSL Connection

If the primary port of Integration Server is an HTTPS port with the client authentication set to **Require Client Certificate**, perform the following configuration steps in Integration Server to ensure a successful SSL connection.

These configuration settings ensure that the Platform Manager plug-in for Integration Server uses the certificates corresponding to **Keystore Alias** and **Key Alias** that are configured in the HTTPS ports and send the same certificates to Integration Server during SSL connection from Command central.

To configure Integration Server for SSL connection

1. Add the certificates to the truststore configured in the Integration Server HTTPS primary port.
2. Map the certificates to an Administrator User in Integration Server.

For information on adding certificates to truststore and mapping certificates to Integration Server user, see [“Authenticating Clients” on page 515](#).

Monitoring Integration Server Instances

Run-time Monitoring Statuses for IntegrationServer-*instanceName*

The following table lists the run-time statuses that the IntegrationServer-*instanceName* run-time component can return in response to the `sagcc get monitoring rntimestatus` and `sagcc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The Integration Server is running, and it is accepting and processing requests over the Integration Server primary port.
PAUSED	The Integration Server is in quiesce mode. Integration Server is accepting or processing requests only over the diagnostic port and the quiesce port.
STOPPED	The Integration Server has been stopped. Integration Server is not accepting nor processing requests over the Integration Server primary port.
UNKNOWN	The status of the Integration Server cannot be determined.
UNRESPONSIVE	The Integration Server is running, but not reachable.

Note:
IS-*instanceName* might still report ONLINE, which indicates there is an issue with Integration Server.

Run-time Monitoring States for IntegrationServer-*instanceName*

In response to the `sagcc get monitoring rntimestate` and `sagcc get monitoring state` commands, IntegrationServer-*instanceName* run-time component provides information about the following key performance indicators (KPIs):

KPI	Description
Running Services	<p>Use this KPI to monitor the number of services that Integration Server is running concurrently so that you can take corrective actions if the number approaches a critical value. The number of running services includes services that were triggered, scheduled, or directly invoked.</p> <p>The KPI uses the following marginal, critical, and maximum values:</p> <ul style="list-style-type: none"> ■ Marginal is 80% of the high water mark of concurrently running services.

KPI	Description
Response Time	<ul style="list-style-type: none"> ■ Critical is 95% of the high water mark of concurrently running services. ■ Maximum is 100% of the high water mark of concurrently running services. This is shown in the Threads area of Integration Server. <p>Use this KPI to monitor service response time so that you can take corrective actions if the response time approaches a critical value.</p> <p>The KPI uses the following marginal, critical, and maximum values:</p> <ul style="list-style-type: none"> ■ Marginal is 80% of the high water mark of service response time. ■ Critical is 95% of the high water mark of service response time. ■ Maximum is 100% of the high water mark of service response time.
Service Errors	<p>Use this KPI to monitor how many service exceptions have occurred in the last minute so that you can take corrective actions if the current number of exceptions is approaching a critical value.</p> <p>The KPI uses the following marginal, critical, and maximum values:</p> <ul style="list-style-type: none"> ■ Marginal is 5 exceptions in the last minute. ■ Critical is 20 exceptions in the last minute. ■ Maximum is more than 20 exceptions in the last minute.

Monitoring KPIs of Integration Server Instances

Perform the following procedure to monitor KPIs of Integration Server instances.

> To view the KPIs of Integration Server instances

1. On the **Environments** pane, select the environment you want to monitor.
2. Click the **Instances** tab.
3. In the table, select the Integration Server you want to monitor.
4. Click the **Overview** tab.

The **Monitoring** section in the **Dashboard** shows the KPIs of the Integration Server instance.

Integration Server returns the following three KPIs:

Name	Marginal Value	Critical Value	Maximum Value
Average response time (in ms)	80% of maximum	95% of maximum	5000
Service errors	70% of maximum	90% of maximum	5
Running services	80% of maximum	95% of maximum	10

Integration Server Configuration Types

Configuration Types that `IntegrationServer-instanceName` Supports

`IntegrationServer-instanceName`, where *instanceName* is the name of the Integration Server instance, run-time component supports creating instances of the configuration types listed in the following table.

Configuration Type	Configuration ID	Use to configure...
Admin UI Link	COMMON-ADMINUI	Full URL to Integration Server.
Clustering	COMMON-CLUSTER	Configuration instance to enable or disable a cluster configuration and to specify cluster settings such as the Terracotta Server array URLs, session timeout, and Action On Startup Error.
Database functional aliases	COMMON-DBFUNCTION	Configuration instance for database functional aliases for Integration Server.
JDBC connection pools	COMMON-JDBC	Configuration instance for JDBC connection pools for Integration Server.
JMS settings	COMMON-JMS	Configuration instance for JMS settings for Integration Server.
JNDI settings	COMMON-JNDI	Configuration instance for JNDI settings for Integration Server.
Keystore and truststore aliases	COMMON-KEYSTORES	Configuration instance for a KeyStore alias that identifies a keystore file or private key within a keystore.
LDAP directories	COMMON-LDAP	Configuration instance for LDAP directory settings for Integration Server.

Configuration Type	Configuration ID	Use to configure...
		<p>Each LDAP connection must be declared using a separate COMMON-LDAP configuration instance.</p> <p>Note: Currently, only one configuration instance, COMMON-LDAP-default is supported per Integration Server.</p>
Integration Server Core and Terracotta license files	COMMON-LICENSE	<p>License files. IntegrationServer-<i>instanceName</i> supports configuration instances for the Integration Server Core and Terracotta license files.</p>
Integration Server Core and Terracotta license files	COMMON-LICLOC	<p>Locations where license files reside in file system where Integration Server is installed. IntegrationServer-<i>instanceName</i> supports configuration instances for the location of the Integration Server Core license file and for the location of the Terracotta license file.</p> <p>You cannot change the location of the license file.</p>
User management	COMMON-LOCAL-USERS	<p>Configuration instance to manage local users and their passwords. COMMON-LOCAL-USERS-<i><userId></i> supports configuring the various users. Each user must be declared using a separate configuration instance.</p> <p>By default, there are four predefined COMMON-LOCAL-USERS configuration instances, namely, COMMON-LOCAL-USERS-Administrator, COMMON-LOCAL-USERS-Default, COMMON-LOCAL-USERS-Developer, and COMMON-LOCAL-USERS-Replicator.</p> <p>Note: Upon creation, a user is automatically added to the Everybody group.</p>

Configuration Type	Configuration ID	Use to configure...
Integration Server loggers and server log facilities	COMMON-LOGGERS	<p>Configuration instance for Integration Server loggers and server log facilities.</p> <p>For Server Logger, the default value for Levels of Loggers is <code>Inherit</code>, which indicates that the components will have the same logging levels as its parent. You cannot add or delete Logger Levels and can only modify the Level values.</p>
Ports	COMMON-PORTS	<p>Ports. <code>IntegrationServer-instanceName</code> supports configuration instances for HTTP, HTTPS, FTP, FTPS, and Diagnostics ports.</p>
Proxy server aliases	COMMON-PROXY	<p>Configuration instance for proxy server aliases.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: You can use the default proxy alias, proxy-bypass, to optionally route selected requests directly to their targets, bypassing the proxy. The proxy bypass addresses list applies to all proxy server aliases. You cannot create or delete the proxy-bypass alias.</p> </div>
Email settings	COMMON-SMTP	<p>Settings for sending email messages.</p>
Server configuration parameters	IS-SYSPROPS	<p>Server configuration parameters. <code>IntegrationServer-instanceName</code> supports appending, updating, and removing system configuration parameters defined in the <code>server.cnf</code> configuration file:</p> <p><i>Integration Server_directory</i> <i>/instances/instance_name/config/server.cnf</i>, where <i>instance_name</i> is the name of the Integration Server instance.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: IS-SYSPROPS is supported only through Command Central template (YAML).</p> </div>

Configuration Type	Configuration ID	Use to configure...
	COMMON-SYSPROPS	<p>Server configuration parameters. <i>IntegrationServer-instanceName</i> supports overwriting the system configuration parameters defined in the <i>server.cnf</i> configuration file:</p> <p><i>Integration Server_directory</i> /instances/<i>instance_name</i>/config/server.cnf, where <i>instance_name</i> is the name of the Integration Server instance.</p> <p>Note:</p> <p>Both Command Central user interface and Command Central template support COMMON-SYSPROPS.</p>
Keystore and truststore aliases	COMMON-TRUSTSTORES	Configuration instance for a TrustStore alias that identifies a truststore file.
Global variables	COMMON-VARS	Configuration instance for global variables. Each variable must be declared using a separate configuration instance.
webMethods Messaging setting	COMMON-WMMESSAGING	<p>Configuration instance for webMethods Messaging settings for Integration Server. <i>IntegrationServer-instanceName</i> supports configuration instances for webMethods Messaging providers (webMethods Broker and Universal Messaging).</p> <p>Note:</p> <p>After creating the Broker connection alias from Command Central with connection state enabled as true, restart Integration Server for the changes to take effect.</p>
Web service endpoint alias - Consumer	IS-CONSUMER-ENDPOINTS	Consumer web service endpoint aliases.
webMethods Messaging setting	IS-DEFAULT-WMMESSAGING	The default messaging connection alias.

Configuration Type	Configuration ID	Use to configure...
Database functional aliases	IS-FILEPERMISSION	File access control configuration for the pub.file services in WmPublic package. The IS-FILEPERMISSION configuration instance includes IS-FILEPERMISSION_READ, IS-FILEPERMISSION_WRITE, and IS-FILEPERMISSION_DELETE. You cannot create a new configuration instance or delete an existing configuration instance. For more information about file access control configuration, see <i>webMethods Integration Server Built-In Services Reference</i> .
Ports	IS-PRIMARYPORT	The primary port ID for Integration Server.
Web service endpoint alias - Provider	IS-PROVIDER-ENDPOINTS	Provider web service endpoint aliases.
Ports	IS-QUIESCEPORT	The quiesce port ID for Integration Server.
Resource settings	IS-RESOURCES	Configuration instance for resource settings for Integration Server. <i>IntegrationServer-instanceName</i> supports configuration instances for outbound HTTP, stateful sessions limit, server thread pool, document stores, and XA recovery store settings.
JAAS realms settings	COMMON-JAAS-REALMS	<p>Configuration instance to perform create, read, update, and delete operation on JAAS realms.</p> <p>Integration Server supports the following predefined JAAS-REALMS configuration instances:</p> <p>IS_KERBEROS_INBOUND</p> <p>IS_KERBEROS_OUTBOUND</p> <p>IS_Transport</p> <p>PlatformManagement</p> <p>WSS_Message_IS</p>

Configuration Type	Configuration ID	Use to configure...
		<p>WSS_Transport_IS</p> <p>WSS_Transport_ProxyService</p> <p>When adding a realm in the custom login module, make sure you use the following format:</p> <pre>customRealm{ \${loginModuleName} Flag; };</pre> <p>Note: If the input values are not in a correct format, Integration Server fails to start with the following error: javax.security.auth.login.LoginException: No LoginModules configured for \${realm_Name}”.</p>
Groups Management	IS-GROUPS	<p>Configuration instance to manage local groups and their member users on Groups management. LDAP and CDS groups will be listed using this configuration instance, but they cannot be created, edited, or deleted.</p> <p>IS-GROUPS-<GroupType>_<groupName> supports configuring various groups.</p> <p>Each group must be declared using a separate configuration instance. <GroupType> can be “local” corresponding to local groups / “LDAP” corresponding to LDAP groups / “system” corresponding to CDS groups.</p> <p>Integration Server supports the following predefined IS-GROUPS configuration instances:</p> <p>IS-GROUPS-local_Administrators</p> <p>IS-GROUPS-local_Anonymous</p> <p>IS-GROUPS-local_Developers</p> <p>IS-GROUPS-local_Everybody</p>

Configuration Type	Configuration ID	Use to configure...
		IS-GROUPS-local_Replicators.
ACLs Management	IS-ACLS	<p>Configuration instance to manage ACLs and their member groups.</p> <p>IS-ACLS-<aclName> supports configuring the various ACLs. Each ACL must be declared using a separate configuration instance.</p> <p>Integration Server supports the following predefined IS-ACLS configuration instances:</p> <p>IS-ACLS-Administrators</p> <p>IS-ACLS-Anonymous</p> <p>IS-ACLS-Default</p> <p>IS-ACLS-Developers</p> <p>IS-ACLS-Internal</p> <p>IS-ACLS-Replicators</p>
SFTP Server	IS-SFTP-SERVERALIAS	Configuration instance for SFTP server settings in Integration Server.
SFTP User	IS-SFTP-USERALIAS	Configuration instance for SFTP user settings in Integration Server.

Working with Integration Server Configuration Types

Perform the following procedure to add, edit, or delete items for Integration Server configuration type items over Command Central.



Note:

Ensure that Integration Server is running before performing the following procedure.

> To add, edit, or delete an item for an Integration Server configuration type

1. Select the Integration Server environment from the Environment pane, then click the instance from the **Instances** tab.
2. Click the **Configuration** tab.
3. Select the configuration type from the drop-down list.

Command Central displays the available or default values, if any for the selected Integration Server configuration type.

4. To add an item for the Integration Server configuration type, click . Enter the required values in the displayed fields and click **Save**.
5. To edit an item for a configuration type, click on the item that you want to update and click **Edit**. Make the necessary changes and click one of the following:
 - **Test** to test the configuration type item.
 - **Save** to save your changes.
 - **Cancel** to cancel the edits to the configuration type item.
6. To delete an item for a configuration instance, click .

Lifecycle Actions for Integration Server

The following table lists the actions that Integration Server supports with the `sagcc exec lifecycle` command and the operation taken against Integration Server when an action is executed. These actions can also be performed through the Command Central web user interface. .

Action	Description
start	Starts an Integration Server instance.
stop	Stops an Integration Server instance.
restart	Restarts an Integration Server instance.
pause	Sets an active Integration Server in to quiesce mode. Integration Server waits for 1 minute before entering quiesce mode to ensure that in-flight services are complete and packages are not disabled immediately. The Integration Server run-time status is set to PAUSED. When an Integration Server is paused, all ports except the diagnostic port and the quiesce port are disabled. In quiesce mode, any requests that are already in progress are permitted to finish, but any new inbound requests to the server are blocked. Outbound connection attempts, such as connections to JDBC pools or connections through LDAP or a central user directory, remain open.
resume	Switches an Integration Server in quiesce mode to active mode and resumes normal operation. All the assets and activities that were disabled or suspended are restored or resumed. The Integration Server run-time status returns to ONLINE.

Commands that Integration Server Supports

Integration Server supports the following Platform Manager commands:

- `sagcc create configuration data`
- `sagcc delete configuration data`
- `sagcc get configuration data`
- `sagcc update configuration data`
- `sagcc get configuration instances`
- `sagcc list configuration instances`
- `sagcc get configuration types`
- `sagcc list configuration types`
- `sagcc exec configuration validation create`
- `sagcc exec configuration validation delete`
- `sagcc exec configuration validation update`
- `sagcc create instances`
- `sagcc delete instances`
- `sagcc list instances supported products`
- `sagcc update instances`
- `sagcc get inventory components`
- `sagcc list inventory components`
- `sagcc exec lifecycle`
- `sagcc get monitoring`
- `sagcc list administration`

For more information on these commands, see *Software AG Command Central Help Guide*.

Using Unix Shell Scripts to Change Connection Credentials for Managed Products

You can use the following sample Unix shell script to configure basic authentication credentials for product components managed by Command Central.

```
NODE_ALIAS=local
USERNAME=Administrator
PASSWORD=secret
RCID=integrationServer-default

sagcc get configuration data $NODE_ALIAS $RCID COMMON-LOCAL-USERS-Administrator

-o administrator.xml
```



```
sed "s,/>, ><Password>${PASSWORD}</Password></User>,g" administrator.xml >
  administrator_new.xml
sagcc update configuration data $NODE_ALIAS $RCID COMMON-LOCAL-USERS
-Administrator -o administrator_new.xml

# verify connection
sagcc get monitoring runtimestatus $NODE_ALIAS $RCID -e ONLINE
```

Command Central Command Line Tool Upgrade

In all existing Integration Server automation scripts and queries that use the Command Central REST API or command line tool commands, change the Integration Server instance ID for all Integration Server instances from:

```
runtimeComponent=integrationServer-ENGINE
```

to

```
runtimeComponent=integrationServer-instanceName
```

where *instanceName* is the name of the Integration Server instance, for example `integrationServer-default`.

A Integration Server Deployment Checklist

■ Introduction	1008
■ Stage 1: Installation	1008
■ Stage 2: Basic Configuration	1009
■ Stage 3: Setting Up Users, Groups, and ACLs	1010
■ Stage 4: Publishing Packages	1011
■ Stage 5: Installing Run-Time Classes	1012
■ Stage 6: Preparing Clients for Communication with the Server	1012
■ Stage 7: Setting Up Security	1013
■ Stage 8: Startup and Test	1014
■ Stage 9: Archive Sources	1015

Introduction

This appendix contains a useful checklist for setting up your webMethods Integration Server. It describes the steps to perform to put an Integration Server into production. The process is comprised of several stages. You should complete one stage before advancing to the next.

Stage 1: Installation

Complete the following steps to install, run, and test the Integration Server.

Step	Action	Done?
1.	Install the Integration Server. For instructions, see <i>Installing Software AG Products</i> . Note: You can install the Integration Server as either a Windows application or a Windows service. After installation, if you want, you can switch from a Windows application to a Windows service, or vice versa. For instructions, see “Running Integration Server as a Windows Application vs. a Windows Service” on page 44.	
2.	Change default passwords. Use the Integration Server Administrator to assign new passwords to the following user accounts: <ul style="list-style-type: none">■ The "Administrator" user account.■ The "Developer" user account.■ The "Central" user account.■ The "Replicator" user account. For instructions on how to change passwords, refer to “Adding an Administrator User” on page 88. Use the Integration Server Administrator to assign a new master password for the Integration Server to use when encrypting outbound passwords before storing them. For instructions on changing the master password, refer to “Changing the Master Password” on page 540.	
3.	Determine a strategy for outbound passwords and the master password. Before you launch and configure your Integration Server the first time, determine how you want the Integration Server to handle the outbound passwords and master password with respect to where they are stored, how they are encrypted, and how often they must be changed. If you	

Step	Action	Done?
	change these settings after the Integration Server has been configured, the master password and outbound passwords can become out of sync. See “Configuring Integration Server for Secure Communication” on page 457 for more information.	
4.	<p>Determine whether to use a JDK or JRE and specify the Java location.</p> <p>If you intend to use Designer and/or Developer to develop and compile Java services on Integration Server, specify the location of the JDK. If you will not be using this installation of Integration Server to compile Java services, you can specify the location of a JRE or a JDK. For information about specifying the Java location, see “Passing Java System Properties to Integration Server” on page 51.</p>	

Stage 2: Basic Configuration

Use the Integration Server Administrator to configure the way in which the server will send outbound requests, accept inbound requests, expire sessions, and issue error messages.

Step	Action	Done?
1.	<p>Set up the ports.</p> <p>Use the Ports screen to specify the ports on which the server will listen for requests.</p> <p>If you will receive HTTP and/or HTTPS requests on multiple ports, you may want to disable all but one port (the one you will use to interact with the Integration Server Administrator) until the server is ready for production.</p> <p>For instructions on how to set up and disable ports, see “Setting Up Aliases for Remote Integration Servers” on page 115.</p>	
2.	<p>Specify the proxy servers.</p> <p>Use the Proxy Servers screen to specify the proxy server(s) (if any) through which this server will issue outbound requests.</p> <p>Specify which URLs (if any) can bypass the proxy server.</p> <p>For instructions on how to specify proxy servers and bypass lists, see “Setting Up Aliases for Remote Integration Servers” on page 115.</p>	
3.	<p>Configure session timeouts.</p> <p>Use the Resources screen to set the timeout value you want the server to use.</p> <p>For instructions, see “Managing Server Sessions” on page 110.</p>	

Step	Action	Done?
4.	<p>Specify the error message recipients and an SMTP server.</p> <p>Use the Logging screen to specify the e-mail address where you want the server to send error messages when an exception (a critical server error or a binding failure) occurs and the name of the SMTP server to use for this purpose.</p> <p>For instructions, see “Configuring Where the Integration Server Writes Logging, Status, and Other Information” on page 127.</p>	
5.	<p>Set up logging.</p> <p>For instructions, see the <i>webMethods Audit Logging Guide</i> and “Setting Up the Server Log” on page 213.</p>	

Stage 3: Setting Up Users, Groups, and ACLs

Use the Integration Server Administrator to identify user accounts, groups, and access control lists (ACLs) to provide appropriate levels of access to the services that will run on this server.

Step	Action	Done?
1.	<p>Identify service security requirements.</p> <p>Services are implicitly blocked from access by anyone other than Administrators and Developers. Determine what level of access is required, whether limited to one group of users, all authenticated users, or even unauthenticated users, and apply the appropriate ACL to the service.</p>	
2.	<p>Create user IDs and groups or configure an external directory.</p> <p>If you have secure services, identify users and/or client applications that are authorized to access those services and create groups that contain the authorized members.</p> <p>If your site uses an external directory (LDAP or central user management), you can configure the server to access the user and group information from the external directory.</p> <p>For instructions for creating user IDs, see “Adding User Accounts” on page 86. For instruction for creating groups, see “Adding Groups” on page 100. For instructions for using an external directory, see “Configuring a Central User Directory or LDAP” on page 635.</p>	
3.	<p>Create ACLs.</p> <p>Create the ACLs needed to meet your services' security requirements and assign the groups you have created to these ACLs. For instructions, see “Creating ACLs” on page 506.</p>	

Step	Action	Done?
4.	<p>Identify backup administrators.</p> <p>Select one or two users who can act as a backup administrator when the primary administrator is unavailable. Use the Users and Groups screen to add these users to the “Administrators” group.</p> <p>For instructions on how to grant a user administrator privileges, see “Adding an Administrator User” on page 88.</p>	

Stage 4: Publishing Packages

Install and configure the packages that will run on this server.

Step	Action	Done?
1.	<p>Install services on the server.</p> <p>Use one of the following methods to publish your services to the production server:</p> <ul style="list-style-type: none"> ■ <i>Method 1.</i> Use the Packages > Publishing screen to replicate the packages from the development server to the production server. <p>For instructions, see “Copying Packages from One Server to Another” on page 673.</p> ■ <i>Method 2.</i> Use the Integration Server Administrator of the publishing server to create a zip file containing each package you want to publish; then: <ol style="list-style-type: none"> 1. Copy the zip file to following directory on the target server: <p style="margin-left: 40px;"><i>Integration Server_directory \instances\instance_name\replicate\inbound</i></p> 2. Use the Packages > Management screen to install each package. 	
2.	<p>Configure the services on the server.</p> <p>Ensure that each service is enabled. Then, configure the following operating parameters for each:</p> <ul style="list-style-type: none"> ■ ACL assignment <p>For instructions, see “Assigning ACLs to Folders, Services, and Other Elements” on page 509.</p> ■ Caching parameters <p>For instructions, see <i>webMethods Service Development Help.</i></p> ■ Output Template Assignment 	

Step	Action	Done?
	For instructions, see <i>webMethods Service Development Help</i> and the <i>Dynamic Server Pages and Output Templates Developer's Guide</i> .	

Stage 5: Installing Run-Time Classes

If your services use run-time classes beyond those provided by Java or the Integration Server (e.g., CORBA or MQ Series classes), install those classes on the server.

Step	Action	Done?
1.	<p>Install run-time classes.</p> <p>Obtain the zip or jar file from the vendor, and then copy the zip or jar file to a device or directory that your Integration Server can access.</p>	
2.	<p>Update the classpath.</p> <p>Update the classpath statement in the <code>wrapper.java.additional</code> property of the <code>custom_wrapper.conf</code> file so that it points to the directory in which you have installed the run-time classes.</p> <p>For Microservices Runtime, update the Java system properties in <code>setenv.bat/sh</code>.</p> <p>For more information about updating the class path, see “Adding Classes to the Server Classpath” on page 917.</p>	

Stage 6: Preparing Clients for Communication with the Server

If you have applications (for example, Java or C/C++ programs) that you want to be Integration Server clients, you must prepare the clients for communication with Integration Server.

Step	Action	Done?
1.	<p>The Integration Server <code>client.jar</code> file contains classes that clients need to communicate with Integration Server. If you have clients on the same machine as Integration Server or Designer, set the classpath on the machine to include the <code>wm-isclient.jar</code> file. The <code>wm-isclient.jar</code> file is located in the <code>common\lib</code> directory, so set the classpath to <code>%CLASSPATH%; Software AG_directory\common\lib\wm-isclient.jar</code>.</p>	
2.	<p>If you have clients on machines that do not also host either Integration Server or Designer, do the following for each machine:</p> <ol style="list-style-type: none"> 1. Navigate to the <code>Software AG_directory\common\lib</code> directory and copy the <code>wm-isclient.jar</code> file to any directory on the client machine. 	

Step	Action	Done?
	<ol style="list-style-type: none"> If you want the client to use SSL to communicate with Integration Server, navigate to the <i>Software AG_directory</i> \common\lib\ext directory and copy the enttoolkit.jar file to any directory on the client machine. Set the classpath on the client machine to include the wm-isclient.jar file and, if applicable, the enttoolkit.jar file. For example, if you put the wm-isclient.jar file and the enttoolkit.jar file in the c:\myapp directory, you would set the classpath to %CLASSPATH%;c:\myapp\client.jar;c:\myapp\enttoolkit.jar. 	

Stage 7: Setting Up Security

Take the following steps to ensure that the security measures you want to use are in place.

Step	Action	Done?
1.	<p>Check passwords.</p> <p>Verify that the passwords for the Administrator and Replicator accounts and the master password for outbound password encryption have been changed from the default values assigned by webMethods Integration Server.</p>	
2.	<p>Edit the index.html file to prevent access to Integration Server Administrator.</p> <p>If you want to prevent a user from inadvertently accessing the Integration Server Administrator, edit the following file:</p> <p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\packages\Default\pub\index.html</p> <p>Change the SRC in the <frame src="/WmRoot/index.dsp"> tag to some innocuous page you have created (perhaps one that displays an error message with alternate links).</p> <p>You can prevent users from seeing all DSP files on Integration Server by using the watt.server.displayDirectories server configuration parameter.</p> <p>Note that if you edit the index.html file, you will not be able to invoke the Integration Server Administrator in the standard way (i.e., simply connecting to the Integration Server's listening port). Instead, you will need to type the Integration Server Administrator's complete URL as shown below:</p> <pre>http://Server:Port /WmRoot/index.dsp</pre> <p>where:</p> <p><i>Server</i> is the name of the Integration Server, and</p> <p><i>Port</i> is the port on which it listens for HTTP requests.</p>	

Step	Action	Done?
3.	Check user accounts. Verify that all user accounts have passwords as required.	
4.	Check ACL assignments. Verify that all secure services have correct ACL assignments.	
5.	Check proxy server settings. Verify that proxy server settings and bypass list are correct.	
6.	Restrict access. Configure allow/deny lists to restrict inbound requests as necessary.	
7.	Install and configure digital certificates.	
8.	Configure HTTP routing systems. If your server sits behind a routing, load-balancing, or URL-filtering system, consult with the administrator of that system to ensure that it will pass inbound requests to the Integration Server.	
9.	Configure write permissions for specific administrator users to the maskSessionID.properties file. Provide only specific administrator users with write permissions to the maskSessionID.properties file. The remaining administrator users must have only read permissions to the particular file.	
10.	Ensure security of operating system. The security of your Integration Server depends on the security of your operating system. Therefore, make sure your operating system is properly configured, that all security patches have been applied, and that unnecessary network services, such as telnet or mail, have been removed.	

Stage 8: Startup and Test

Start the server and test services to ensure that they operate as expected.

Step	Action	Done?
1.	Verify that ports are enabled. If you disabled the ports to prevent access to the server during the setup phase, use the Ports screen to enable them now.	
	Tip:	

Step	Action	Done?
	After you enable a port, ping it to verify that it is operational.	
2.	<p>Restart the server.</p> <p>Use the Integration Server Administrator to restart the server to ensure all settings that you have made are in effect.</p> <p>For instructions, see “Restarting the Integration Server” on page 55.</p>	
3.	<p>Test services.</p> <p>Perform tests to ensure that user/client applications can access the server successfully.</p> <p>Note: During this test you may also want to verify that your current license will accommodate the expected concurrency demands on this server. Contact Software AG to increase number of licensed sessions if necessary.</p>	
4.	Go Live!	

Stage 9: Archive Sources

Archive a master copy of the packages on the server and the source files that were used to build them.

Step	Action	Done?
1.	Copy the contents of the server\packages directory to another device for backup and archival purposes.	
2.	Archive a copy of all the source files that went into producing the services deployed on this server.	

B Server Configuration Parameters

■	Introduction	1019
■	watt.adapters.	1019
■	watt.adminapi.	1019
■	watt.art.	1020
■	watt.broker.	1021
■	watt.brokerCoder.	1021
■	watt.cachedirective.	1021
■	watt.cds.	1022
■	watt.client.	1022
■	watt.config.	1023
■	watt.core.	1023
■	watt.debug.	1029
■	watt.debug2.	1031
■	watt.frag.	1032
■	watt.infradc.	1032
■	watt.net.	1032
■	watt.security.	1048
■	watt.server.	1053
■	watt.ssl.	1177

■ watt.ssh.	1178
■ watt.tx.	1179
■ watt.um	1180
■ watt.wm.tnextdc.	1181
■ watt.wmcloud.	1181

Introduction

This appendix contains a description of the parameters you can specify in the server configuration file (server.cnf), which is located in the *Integration Server_directory \instances\instance_name\config* directory. Typically you will use the **Settings > Extended** screen from the Integration Server Administrator to update this file, but there might be times when you need to edit the file directly using a text editor. If you edit the file directly, you should first shut down the Integration Server before updating the file. After you make the changes, restart Integration Server.

Note:

If you are using the **Settings > Extended** screen to update the server configuration file (server.cnf), server restart is not required unless otherwise specified.

Integration Server uses default values for many of the parameters. If a parameter has a default, it is listed with the description of the parameter. Many of these parameters are set as you administer the Integration Server using the Integration Server Administrator.

watt.adapters.

watt.adapters.withOnlineHelp

Specifies a comma-separated list of the adapter names that have online help content that needs to be accessible through the Help link on Integration Server Administrator. There is no default value for this parameter.

watt.adminapi.

watt.adminapi.group.readOnly

Specifies the name of the user group whose members have read-only access to the Integration Server Administrator API. Members can only access the Administrator API using GET requests. User group name are case-sensitive. There is no default value for this parameter.

Note:

Reload the WmAdmin package for changes to this parameter take effect. Alternatively, restart Integration Server.

watt.adminapi.log.clientErrors

Specifies whether client errors that occur during execution of REST resources in the Administrator API appear in the error log. Set this parameter to true to write client errors, those that result in an HTTP status code in the 400s, to the error log. Set this parameter to false if you do not want client errors to be written to the error log. The default is false.

watt.adminapi.returnExceptions

Specifies whether the HTTP response sent when the Integration Server Administrator encounters an exception includes a stack trace in the response body. Set this parameter to true to include the stack trace in the response body in addition to the exception message. Set this parameter to false to omit the stack trace and return only the exception message. The default is true.

watt.art.

Note:

For additional server configuration parameters related to adapters, see the adapter documentation.

watt.art.analysis

Specifies whether to enable logging to analyze adapter listeners and their linked notifications. When set to true, `wm.art.dev.notification:analyzeListenerNotifications` service and `wm.art.dev.listener:analyzeListenerNodes` service will log the data in the server log file. Service `wm.art.dev.listener:updateRegisteredNotifications` will update the listener for missing listener notifications. When set to false, no analysis is done and no information is logged. The default is false.

watt.art.connection.nodeVersion

Specifies whether the adapter connection stores the password in the passman store and the password handle in the connection node. When the `watt.art.connection.nodeVersion` parameter is set to 1, the password is embedded in the adapter connection. When the parameter is set to 2, the password handle is stored in the adapter connection. The default is 2. Software AG recommends using the default value. Every time you set a new value for this parameter, you must restart Integration Server or reload the WmART package.

When the value of the `watt.art.connection.nodeVersion` parameter is 2, during runtime-based deployment with Deployer you must perform variable substitution for the password field to deploy the password to the target system.

watt.art.notification.publishToJMS.useCSQ

Specifies whether the asynchronous notification of Adapter Runtime can use the client side queue (CSQ) when publishing to the JMS provider. When set to true, the asynchronous notifications being sent to the JMS provider can be published to the CSQ when the JMS provider is unavailable. The default is false.

watt.art.page.size

Specifies the maximum number of items to be displayed on an adapter's Connections screen, Listeners screen, and Notifications screen. The default is 10. For more information about controlling pagination, see the adapter documentation.

watt.art.synchronousNotification.selectExecuteUser

Specifies WmArt-based adapters that are to include a **Run as User** column in the Listener Notifications screen. With this column in place, you can assign a user to a notification. Then, when the listener notification invokes a service, it runs as the specified user. You can specify one or more adapters. If you specify multiple adapters, separate the names with semicolons (;), for example:

```
watt.art.synchronousNotification.selectExecuteUser=WmMQAdapter;WmSAP
```

watt.art.service.pipeline.hidden

Specifies whether the adapter service pipeline is logged in the Integration Server log file. When the `watt.art.service.pipeline.hidden` parameter is set to `true`, the service pipeline is not logged in the Integration Server log file. When the parameter is set to `false`, the service pipeline is logged in the Integration Server log file. The default is `false`.

watt.broker.

watt.broker.sync.enableBrokerSync

Indicates whether Integration Server synchronizes to the Broker client. If this property is set to `true` (the default), then Integration Server synchronizes to the Broker client. If the value is `false`, Integration Server does not sync to the Broker client and logs a warning message indicating that the sync is turned off.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.broker.sync.forceDispatcherInit

Indicates whether Integration Server should reinitialize the Broker configuration. If this value is set to `true`, Integration Server reinitializes the Broker configuration. Integration Server reinitializes from the configuration file if available. If the configuration file is unavailable, Integration Server reinitializes the Broker configuration with default values. The default is `false`.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.brokerCoder.

watt.brokerCoder.verbose

Indicates whether Integration Server enables verbose logging for BrokerCoder. If the value is set to `true`, Integration Server is enabled for BrokerCoder verbose logging. The default is `false`.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.cachedirective.

watt.cachedirective.exclude.packages

Specifies a comma-separated list of packages whose Dynamic Server Pages you want the browser to cache. You can specify the packages as regular expressions. The asterisk (*) is the only wildcard

character allowed in the regular expression. By default, the value of this parameter is empty, meaning none of the Dynamic Server Pages in the Integration Server Administrator are cached.

Note: Software AG recommends that you use `watt.cachedirective.exclude.packages` to cache the Dynamic Server Pages that are related to custom packages only.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.cds.

watt.cds.skip.role.types

Controls whether Integration Server evaluates LDAP or database query roles in the server. If LDAP or database query roles are not used for any ACL management in Integration Server with Common Users enabled, evaluation of these roles might impact Integration Server performance. In this case, you can disable the query roles evaluation function to increase performance. For more information, see [“Considerations for My webMethods Server Query Roles” on page 640](#).

To disable LDAP query role evaluation, set `watt.cds.skip.role.types=wm_xt_ldapqueryprovider`.

To disable database query role evaluation, set `watt.cds.skip.role.types=wm_xt_dbrole`.

To disable both query roles, enter the extended setting twice using both settings above. For example, enter the following on the **Edit Extended Settings** page:

```
watt.cds.skip.role.types=wm_xt_ldapqueryprovider
```

```
watt.cds.skip.role.types=wm_xt_dbrole
```

To re-enable evaluation, delete either or both of the extended settings. If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.client.

watt.client.ftp.session.logoutOnServiceCompletion

Specifies whether an FTP(S) session created by the `pub.client.ftp:login` service closes automatically or not, when the top-level service execution completes. Set the value to `true` to automatically close the FTP(S) session created by the `pub.client.ftp:login` service, when the top-level service execution completes. If the value is set to `false`, you must invoke the `pub.client.ftp:logout` service to close the FTP(S) session. The default value is `false`.

watt.client.sftp.session.logoutOnServiceCompletion

Specifies whether an SFTP session created by the `pub.client.sftp:login` service closes automatically or not, when the top-level service execution completes. Set the value to `true` to automatically close the SFTP session created by the `pub.client.sftp:login` service, when the top-level service execution completes. If the value is set to `false`, you must invoke the `pub.client.sftp:logout` service to close the SFTP session. The default value is `false`.

Note:

The `watt.client.ftp.session.logoutOnServiceCompletion` and `watt.client.sftp.session.logoutOnServiceCompletion` parameters are introduced for PIE-66690 in IS_10.5_Core_Fix8.

watt.config.

watt.config.systemProperties

Specifies the list of additional system parameters whose name does not start with `watt`. Each additional system property is separated by a comma. By default, the property `mail.imap.partialfetch` is included as an additional system property with a default value set to `true`.

watt.core.

watt.core.brokerTypeCoder.verbose

Enables or disables verbose logging for `BrokerTypeCoder`. If set to `true`, Integration Server enables verbose logging for `BrokerTypeCoder`. The default is `false`.

Note:

This parameter is deprecated because `webMethods Broker` is deprecated.

Important:

You must restart Integration Server after you modify the value of this property.

watt.core.brokerCoder.wireFormat

Specifies the wire type Integration Server uses to encode and decode Broker messages. The default is 3.

The following table lists the possible settings:

When set to	Integration Server
0	Drops the current reference from the Broker context only if the tag information is invalid and continues further encoding.
1	Always drops the current reference from the Broker context without verifying whether the tag information is valid and then continues further encoding.
2	Uses the default values specific to Broker for encoding and decoding Broker messages.
	<p>Note:</p> <p>If set to 2 or 3, Integration Server sets the "is_Surrogate" flag to true to enhance Broker message encoding. By default, if the "is_Surrogate" flag is set, Broker uses the <code>BROKER_INT</code> wire type for encoding and decoding, which is the Broker default.</p>
3	Uses current values set in Broker for encoding and decoding Broker messages.

When set to **Integration Server****Note:**

If set to 2 or 3, Integration Server sets the "is_Surrogate" flag to true to enhance Broker message encoding. By default, if the "is_Surrogate" flag is set, Broker uses the BROKER_INT wire type for encoding and decoding.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

Important:

You must restart Integration Server after you modify the value of this property.

watt.core.datatype.patternMatcherPool.delayFill

Indicates whether Integration Server should initialize the pool with minimum pattern matcher pool objects specified. If set to `false` (the default) Integration Server initializes the pool whenever it parses or validates a schema. If the `delayFill` is set to `true`, Integration Server *does not* initialize the pool.

Important:

You must restart Integration Server after you modify the value of this property.

watt.core.datatype.patternMatcherPool.maxSize

Specifies the maximum number of pattern matcher pool objects Integration Server allows in a pool. The default is 50.

watt.core.datatype.usejavaregex

Specifies whether Integration Server uses the Java regular expression compiler during validation. When `watt.core.datatype.usejavaregex` is set to `true`, during XML validation, Integration Server uses the Java regular expression compiler and Integration Server performs pattern matching as described by `java.util.regex.pattern`. When set to `false`, during XML validation, Integration Server uses the Perl regular expression compiler and uses pattern matching. The default is `false`.

Note:

The `watt.core.datatype.usejavaregex` parameter affects XML validation, including validation performed by the `pub.schema.validate` service and validation of web service requests and responses performed by Integration Server. The `watt.core.datatype.usejavaregex` parameter also affects schema validation that Integration Server performs when creating an IS schema or IS document type from an XML Schema definition or when creating a web service descriptor from a WSDL document that contains or references a schema definition.

watt.core.generatedTypeName.namespaceName.authority

Specifies the namespace element Integration Server uses to create a service URI. Specify a String value that indicates the hostname. The default is `localhost`.

watt.core.generatedTypeName.localName.prefix

Specifies the prefix Integration Server uses for a type name. This is appended to the type name to form a local name. The default is `__` (two underscores).

watt.core.generatedTypeName.namespaceName.authority

Used to create a service URI, in the format - 'http://' authority path '/' service-name, authority stands for the value specified by this property. String value usually indicating the hostname.

watt.core.schema.createSchema.omitXSDAny

When generating the schema definition in a WSDL, indicates whether Integration Server omits the `xsd:any` element in the complex type definition that corresponds to an open document. A document variable is considered open when the **Allow unspecified fields** property is set to true. When `watt.core.schema.createSchema.omitXSDAny` is set to true, the `xsd:any` element is omitted from the schema portion of the WSDL document even if the document variable has the **Allow unspecified fields** property is set to true. When `watt.core.schema.createSchema.omitXSDAny` is set to false, Integration Server includes the `xsd:any` element in the complex type definition only if the corresponding document variable has the **Allow unspecified fields** property is set to true. The default for `watt.core.schema.createSchema.omitXSDAny` is true.

Note:

For individual web service descriptors, the value of the **Omit xsd:any from WSDL** property overrides the value of the `watt.core.createSchema.omitXSDAny` server configuration parameter. For more information about using the **Omit xsd:any from WSDL** property, see the *webMethods Service Development Help*.

watt.core.schema.anonymousCyclicExtensionDepth

Specifies the nesting level when creating an IS schema from an XML schema definition that contains an anonymous complex type definition that references its own parent complex type extension. The default value is 3, which stops the expansion at the third nesting level. This may be adequate when needing to address the previous, current, or next reference in the hierarchy of an expansion.

watt.core.schema.generateAllTypeDocuments

When generating an IS document type from an XML schema definition, indicates whether or not Integration Server generates IS document types for all complex types regardless of whether the types are referenced or not. When this property is set to `true`, Integration Server generates an IS document type for every complex type definition in the XML schema. When this property is set to `false`, Integration Server generates a separate IS document type for a complex type only if the complex type is referenced or is derived from a referenced complex type. The default is false.

Note:

This parameter is obsolete.

watt.core.schema.generateObsoleteDocumentTypeNames

Specifies whether Integration Server uses the naming convention that existed prior to version 8.2 when generating document types for complex types referenced from a global element declaration. When this parameter is set to true and the **Always generate document types for referenced elements** check box is selected in Designer when creating a document type, Integration Server uses "docType" as the beginning of the name for document types created for referenced elements. When this parameter is set to false, Integration Server uses "docTypeRef" as the beginning of the name for document types created for referenced element. The default is false.

Important: Software AG does not recommend using the naming convention that existed prior to Integration Server 8.2. Use of the old naming convention might interfere with newer features that observe and expect the later naming convention.

watt.core.schema.generateSubstitutionGroups

When generating an IS document type from an XML schema definition that contains a substitution group, indicates whether the resulting document type contains an optional element for each member of a substitution group. When this property is set to `false`, the resulting document type contains a field that corresponds to the head element in the substitution group, but does not contain any elements for members of the substitution group. When this property is set to `true`, the resulting document type contains a field that corresponds to the head element and fields that correspond to each member element of the substitution group. All the fields, including the head element, are marked as optional elements. The default is `false`.

watt.core.schema.useUnboundedForMaxOccurs

Specifies the number at which a `maxOccurs` value greater than this number is treated as unbounded. When validating a document against an IS schema with an element for which `maxOccurs` is greater than the `watt.core.schema.useUnboundedForMaxOccurs`, Integration Server validates the instance document as if the value of the `maxOccurs` were unbounded. Treating `maxOccurs` as unbounded may prevent an `OutOfMemoryError` from occurring during validation when the `maxOccurs` value is large, such as 9999. There is no default value for this parameter, indicating that Integration Server uses the value specified in the schema as the `maxOccurs` value.

Note:

To use this parameter, you must add it to Extended Settings.

watt.core.schema.validateIncomingXSD

This is an internal property. Do not modify.

watt.core.template.enableFilterHtml

Indicates whether Integration Server HTML encodes the output from a flow service and `%value Variable%` tag in a DSP or output template. When this property is set to `true`, the value of `%value Variable%` tags, including XML and JavaScript, is HTML encoded. Having Integration Server HTML encode the values helps prevent cross-site scripting attacks. When this property is `true`, you can still indicate that you do not want Integration Server to HTML encode the output of a `%value Variable%` tag by including the `encode(none)` option (`%value Variable encode(none)%`). When this property is set to `,` the values of `%value Variable%` tags are not encoded, and, as a result, DSP pages or pages resulting from output templates might be vulnerable to cross-site scripting attacks. In case of a flow service, when this property is set to `true`, the output (characters such as `"<"`, `">"`, and so on) from a flow service is encoded. Setting the property to `false` disables the encoding of a flow service output. The default is `true`.

watt.core.template.enableSecureUrlRedirection

For DSP pages, this parameter controls whether Integration Server performs secure `UfalseRL` redirection. Internal DSP pages commonly use the variable names `"url"` and `"returnurl"` with the `%value%` tag to indicate to Integration Server that URL redirection can occur in the client. As a result, whether a DSP page is an internal DSP page or a custom DSP page, when Integration Server encounters the variable name `"url"` or `"returnurl"` for a `%value%` tag, it assumes that the value is meant for URL redirection.

When the `watt.core.template.enableSecureUrlRedirection` parameter is set to `true`, Integration Server uses secure URL redirection. When performing secure URL redirection, if the `%value%` tag uses the variable name `"url"` or `"returnurl"`, Integration Server checks the URL to determine whether it is a relative path to a location within Integration Server (e.g., `..\redirectedurl.dsp`) or another value. If it is a relative path, Integration Server considers a redirection to the URL to be safe, and as a result, takes no action. However, for other values, Integration Server assumes the URL to be an external URL (e.g., `http://example.com`) and alters the output, even if the intent of the `%value%` tag is not to redirect to the URL. Integration Server alters the URL so that if an attempt is made to redirect to the external URL, the redirection goes to an error page. Integration Server alters the URL by prepending the value `"error.dsp?data="` to the output of the `%value%` tag (e.g., `error.dsp?data=http://example.com`).

Setting this parameter to `false` disables secure URL redirection and, as a result, might put your applications at risk.

You should use the variable `"url"` or `"returnurl"` with the `%value%` tag only when you want to redirect to an internal URL. If you have existing applications that use the variable name `"url"` or `"returnurl"`, Software AG recommends that you update your applications and do not use these variable names except for internal URL redirection. If you do not want to change the variable names in existing applications, you can set `watt.core.template.enableSecureUrlRedirection` parameter to `false`.

The default setting for the `watt.core.template.enableSecureUrlRedirection` parameter is `true`.

watt.core.transientStore.logExceptions

Specifies whether or not Integration Server logs the exceptions from the transient store. Some exception conditions thrown by the transient store, such as interruptions and deadlocks, are handled internally and Integration Server recovers from them silently. When set to `true`, Integration Server logs exceptions thrown by the transient store that might cause the consumer thread to terminate. This can aid in diagnosing problems with an internal store during the resubmission process. When set to `false`, Integration Server does not log all exceptions from the transient store. By default, this parameter is set to `false` to prevent false alarms from exceptions that Integration Server handles and recovers from.

Software AG recommends that you set this parameter to `true` only when directed to do so by Software AG Global Support.

watt.core.validation.skipAbsentStarBody

Specifies whether to skip a validation that Integration Server performs when decoding mixed content elements that have an enumeration restriction. The validation returned an error when optional missed content was absent, because an empty value was not listed as an option in the enumeration set for the `*body` field. When set to `true`, Integration Server skips this validation and treats the `*body` as completely optional, allowing empty or absent `*body` values to pass validation.

When set to `false`, Integration Server performs the validation. The default value is `false`.

Changes to this property take effect immediately.

watt.core.validation.multipleroot

Specifies whether the `pub.schema:validate` service is to validate multiple roots when processing multi-part documents. When this property is set to `true`, the `pub.schema:validate` service checks for multiple root nodes. If multiple root nodes are found, the service flags a validation error. When this property is set to `false`, the `pub.schema:validate` service does not perform multiple root validations. The default is `true`.

watt.core.validation.skipMandatoryFields

Specifies whether Integration Server generates errors during document validation if mandatory fields are missing from the document. When `watt.core.validation.skipMandatoryFields` is set to `true`, Integration Server does not generate validation errors during document validation even if required fields are missing from the document. When this parameter is set to `false`, Integration Server generates validation errors if required fields are missing from the document. The default is `false`. This parameter affects document validation performed by the `pub.schema:validate` service and SOAP request and SOAP response validation.

watt.core.validation.skipNoNamespaceReference

Indicates whether or not Integration Server skips validation of references to elements that are not namespace qualified made from within a namespace qualified element during XML validation against a schema. Set to `true` if you want Integration Server to skip validation of these referenced elements. Set to `false` if you want Integration Server to validate these referenced elements. The default is `false`.

watt.core.validation.w3cConformant

Indicates whether Integration Server evaluates instances of data types for illegal values during XML validation according to the W3C recommendation *XML Schema Part 2: Datatypes*. If you want Integration Server to validate data types consistent with the W3C recommendation, set `watt.core.validation.w3cConformant` to `true`. The default is `false`.

watt.core.xml.allowedExternalEntities

Identifies a list of trusted external entities (file URIs, HTTP URLs, and so on) that contain XML content to transform in an XML or XSLT file. Use a comma to separate each entity.

The `pub.xslt.Transformations:transformSerialXML` service loads the content for any entity in this list that is referenced either in the XML that the service receives or in the XSLT stylesheet the service uses to transform the XML. The service reads this list in the following situations:

- The service's `loadExternalEntities` input parameter is set to `false`.
- The service's `loadExternalEntities` input parameter is not specified in the service signature, and the `watt.core.xml.expandGeneralEntities` server parameter is set to `false`.

watt.core.xml.expandGeneralEntities

Indicates whether the `pub.xml:loadXMLNode` and `pub.xml:xmlStringToXMLNode` services should return expanded general (internal and external) entities. Also indicates whether the `pub.xslt.Transformations:transformSerialXML` service should allow external entities. When this parameter is set to `true`, the `loadXMLNode` and `xmlStringToXMLNode` services expand references to general entities, and the `transformSerialXML` service allows external entities. When this parameter is set to `false`, the `loadXMLNode` and `xmlStringToXMLNode` services ignore references to general entities, and the `transformSerialXML` service blocks external entities. The default is `true`.

Note:

This parameter applies to all instances of `loadXMLNode`, `xmlStringToXMLNode`, and any XSLT services. However, if the `expandGeneralEntities` input parameter is specified for the `loadXMLNode` or `xmlStringToXMLNode` service, or if the `loadExternalEntities` parameter is specified for the `transformSerialXML` service, the service-level setting takes precedence over `watt.core.xml.expandGeneralEntities`.

watt.core.xsd.useGeneratedURIForCreateXSD

Indicates whether, while creating the XSD, Integration Server should use the node's path for the URI and the namespace name for the node and schema. If set to `true`, Integration Server generates the URI based on the node's path. This URI is then used as the namespace name for the particular node or schema. If set to `false` (the default), Integration Server sets the namespace to null.

watt.core.xsd.useGeneratedURIForNonRPC

Specifies whether Integration Server should use the URI generated for the message as the namespace name for the WSDL message. If set to `true` (the default), Integration Server uses the URI generated for the message as the namespace name for the WSDL message only if the style is non-RPC. If set to `false`, Integration Server sets the namespace to null.

watt.core.xsd.useKnownSchemaLocation

Specifies whether Integration Server includes a "known schema location" and the `schemaLocation` attribute for an `import` statement in the WSDL document generated for a provider web service descriptor. When the `schemaLocation` attribute is not present in the WSDL document, an Integration Server that consumes the WSDL document will not dereference the `schemaLocation` attribute value during web service processing. If the known schema locations are the only import statements that required namespace resolution, the absence of the `schemaLocation` attribute allows web service development to occur while not connected to the Internet. Set this parameter to `false` to omit the `schemaLocation` attribute in an import statement when the namespaces is one of the following:

`http://www.w3.org/XML/1998/namespace`

`http://www.w3.org/2003/05/soap-envelope`

The default value is `true`.

Note:

Note: WS-I compliance requires that in a WSDL document, or files referenced by a WSDL document, all XML Schema imports must contain a `schemaLocation` attribute. If an XML Schema in the WSDL generated for a web service descriptor contains an import statement for either of the specified namespaces, setting `watt.core.xsd.useKnownSchemaLocation` to `false` breaks WS-I compliance.

watt.debug.

watt.debug.layout

Specifies the format of messages written to the server's log file and to the **Logs > Server** screen. You can specify one of the following formats:

- `new`

Messages will be in the following format:

*(Component) [ComponentID.00SubComponentID.SubComponentID.MessageKey] TimeStamp
MessageType MessageText*

(IS.SERVER) [ISS.0025.25.6] 2007-07-31 10:45:27 EDT INFO: License Manager started

- `legacy`

This format corresponds to the message format used in Integration Server prior to version 7.1. Use this format if you need to maintain backward compatibility with the previous message format. For example, you might have written code to process messages written to the server log.

When you select legacy as the message layout, messages will appear in the following format:

TimeStamp [ComponentID.00SubComponentID.MessageKeyMessageType] MessageText

2007-07-31 10:39:59 EDT [ISS.0025.0006I] License Manager started

This is the default.

watt.debug.level

Sets level of debugging information written to the server's log file and the **Logs > Server** screen. The default is Info.

Specify	To display
Off	No messages.
Fatal	Fatal messages only.
Error	Error and fatal messages.
Warn	Warning, error, and fatal messages.
Info	Informational, warning, error, and fatal messages. This is the default.
Debug	Debug, informational, warning, error, and fatal messages.
Trace	Trace, debug, informational, warning, error, and fatal messages.

Note:

You can also set the value of the `watt.debug.level` property by setting the logging level for the Default facility on the **Settings > Logging > View Server Logger Details** screen. For more information about configuring logging, see [“Specifying Amount and Type of Information to Include in the Server Log” on page 215](#).

Prior to Integration Server 7.1, Integration Server used a number-based system to set the level of debug information written to the server log. Integration Server maintains backward compatibility with this system. The table below describes the number-based system.

Specify	To record
0	Critical messages only.
1	Error and critical messages.
2	Warning, error, and critical messages.
3, 4	Informational, warning, error, and critical messages.

Specify	To record
5, 6, 7	Debug, informational, warning, error, and critical messages. This is the default.
8, 9, 10	Trace, debug, informational, warning, error, and critical messages. The server records more levels of informational messages the higher you set the number.

watt.debug.logfile

Specifies the fully qualified path or relative path to the file to which you want Integration Server to write server log information. The “relative path” is relative to the Integration Server home directory: *Integration Server_directory \instances\instance_name*

You must specify a directory and a filename.

The default is: *Integration Server_directory \instances\instance_name\logs\server.log*

Note:

If you change the setting of this parameter, you must restart Integration Server for changes to take effect.

watt.debug.warnOnClasspathError

Specifies whether or not a warning message about a missing classpath entry will be written to standard out. Set this parameter to `true` to write the message "Classpath entry in ini.cnf not found: <classpath_entry_name>" to standard out. Set this parameter to `false` if you do not want the message written. The default is `false`.

Because this message will be generated before any of the Integration Server logging facilities are initialized, the warning message is written to standard out.

This configuration parameter does not appear in Extended Settings or in the `server.cnf`. To have Integration Server write these warnings, you must pass this property to Integration Server as a command line option. Add a line to *Software AG_directory /profiles/IS_instance/configuration/custom_wrapper.conf*, for example:

```
wrapper.java.additional.NNN=-Dwatt.debug.warnOnClasspathError=true
```

Where *NNN* is the next available `wrapper.java.additional` number.

To write the warning when running Microservices Runtime, use a text editor to specify the property in the `JAVA_CUSTOM_OPTS` parameter in *Integration Server_directory /bin/server.bat(sh)*. For example:

```
set JAVA_CUSTOM_OPTS="-Dwatt.debug.warnOnClasspathError=true"
```

watt.debug2.**watt.debug2.facList**

Specifies a comma-delimited list of enabled facilities for which the server logs information. The facilities are numbered. The default is 999, which indicates the server is to log information for all facilities. Specify 1000 to prohibit the server from logging information for any service.

To view the names of facilities, use the **Log Settings** screen of the Integration Server Administrator to enable and disable facilities for which you want the server to log information

watt.debug2.logstringfile

Specifies the name (without the extension `.txt`) for the dictionary file that contains error codes and facilities. The default is `lib\logstr` (English Version).

watt.frag.

watt.frag.keep.original.servicename

Specifies whether Integration Server retains the original service name for input fields while fragging or compiling a Java service. If this parameter is set to `true`, Integration Server retains the original service name. If set to `false`, Integration Server modifies the original service name. The default is `false`.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.infradc.

watt.infradc.artmonitor

This is an internal parameter. Do not modify.

watt.infradc.artpollinterval

This is an internal parameter. Do not modify.

watt.net.

watt.net.clientKeepaliveAgingLimit

Specifies how long a socket is kept alive, measured in seconds. Before returning a socket to the pool, Integration Server compares the age of the socket connection against the value of the `watt.net.clientKeepaliveAgingLimit`. If the socket is older than the `watt.net.clientKeepaliveAgingLimit` value, then Integration Server does not return the socket to the pool. Instead, Integration Server closes the socket. When another socket connection is needed, Integration Server will create it. If the socket age is less than the `watt.net.clientKeepaliveAgingLimit` value, then Integration Server returns the socket to the connection pool. The default is 180 seconds.

Note:

Even if the connection age is less than the `watt.net.clientKeepaliveAgingLimit` parameter, Integration Server will close the connection if the connection has exceeded the usage limit set by the `watt.net.clientKeepaliveUsageLimit` server configuration parameter.

Note:

The `watt.net.clientKeepaliveAgingLimit` parameter applies only if `watt.net.maxClientKeepaliveConns` is set to a value greater than 0.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.clientKeepaliveTimeout

Controls how long (in seconds) a client keep alive connection can remain idle before Integration Server closes it. The default is 180 seconds (3 minutes).

watt.net.clientKeepaliveUsageLimit

Specifies the maximum number of usages for a socket in a client connection pool. Before returning a socket to the pool, Integration Server compares the number of times the socket has been used to send a request to the `watt.net.clientKeepaliveUsageLimit` value. If the socket usage count is greater than the `watt.net.clientKeepaliveUsageLimit` value, then Integration Server does not return the socket to the pool. Instead, Integration Server closes the socket. If a new socket is needed in the pool, Integration Server creates one. The default value is 100 uses.

Note:

Even if the number of connection usages is less than the `watt.net.clientKeepaliveUsageLimit` parameter, Integration Server will close the connection if the connection has exceeded the age limit set by the `watt.net.clientKeepaliveAgingLimit` server configuration parameter.

Note:

The `watt.net.clientKeepaliveUsageLimit` parameter applies only if `watt.net.maxClientKeepaliveConns` is set to a value greater than 0.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.default.accept

Specifies the default value of the Accept header when an Accept header is not present in the `headers` input parameter to the `pub.client:http` service. If `watt.net.default.accept` does not have a value and no Accept header is present in the `headers` parameter, the `pub.client:http` service does not include an Accept header in the requests it sends. By default, the `watt.net.default.accept` parameter does not have a value.

watt.net.email.validateHost

Controls whether the Integration Server enforces IP access restrictions for e-mail listeners. When defining an e-mail port, you can define IP access restrictions that specify the hosts that are allowed or denied access via the e-mail port. Set this property to `true` if you want server to enforce the IP access restrictions for e-mail listeners or `false` if you do not. The default is `true`.

watt.net.encodeToUpperCase

Specifies whether Integration Server should use uppercase letters when encoding the characters in the request URL. When Integration Server processes a URL request, it converts the characters outside of the ASCII set in the URL to encoded characters by adding "%" followed by two hexadecimal digits. If this parameter is set to `true`, Integration Server uses uppercase letters in the hexadecimal digits. For example, the encoded value for "value1>4" will be "value1%3E4". When set to `false`, Integration Server does not convert the letters to uppercase. For example, the encoded value for "value1>4" will be "value1%3e4". Default is `true`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ftp.ignoreErrors

Specifies, using a comma-separated list, any FTP command error codes that you want the FTP client to ignore. For example, setting the property to "501, 505" causes the FTP client to ignore error codes 501 and 505.

watt.net.ftp.noExtensionKey

Specifies the extension Integration Server should use to determine the MIME type of the input files when the files have no extension. When invoking an Integration Server service through FTP commands, Integration Server uses the specified extension. The default is `ftp_no_extension`, which means that Integration Server cannot determine the MIME type because there is no extension.

watt.net.ftp.securitychecks

Specifies whether or not Integration Server performs FTP security checks when FTP clients access the Integration Server FTP port. Set this property to `true` if you want Integration Server to perform FTP-related security checks. Set this property to `false` if you do not want Integration Server to perform any FTP-related security checks. The default is `true`.

watt.net.ftpClientDataConnTimeout

Specifies the number of milliseconds that a built-in FTP service executing in active mode (as specified by the *transfertype* input parameter) waits for a remote FTP server to connect to it. If the connection is not established in the specified amount of time, an exception is thrown. The default value is 30000 milliseconds (30 seconds).

watt.net.ftpClientTimeout

Specifies the length of time, measured in seconds, an FTP session can be idle before it is removed from memory. The default is 600 seconds (10 minutes).

watt.net.ftpConnTimeout

Specifies the maximum number of milliseconds the FTP listener allows the connection with the client to remain inactive. The default is 15 minutes.

watt.net.ftpDataConn

Specifies whether an Integration Server functioning as an FTP server allows multiple concurrent connections and supports parallel downloads. When this parameter is set to `true`, Integration Server allows parallel downloads and reuses the same FTP session. When this parameter is set to `false`, Integration Server does not allow parallel downloads and reuses the same FTP session. The default is `false`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ftpDataConnTimeout

Specifies the maximum number of milliseconds the FTP listener waits between successive reads when performing a file upload. The default is 60000 milliseconds (60 seconds).

watt.net.ftpPassiveLocalAddr

Specifies the address to be sent by the PORT command. A host name or IP address can be specified.

Note:

This parameter is not applicable when an FTP/FTPS port is bound to an IPv6 address. In that case, the passive mode listen address is the same as the port bind address.

When running in passive mode, the FTP or FTPS port sends a PORT command to the FTP or FTPS client. The PORT command specifies the address and port to which the client should connect to create a data connection. If the FTP or FTPS port is behind a NAT server, however, the address of the host on which the Integration Server runs is not visible to the FTP or FTPS client. Consequently the PORT command does not contain the information the client needs to connect to the server. To remedy this situation, you can specify a value for the `watt.net.ftpPassiveLocalAddr` property.

Alternatively, when you configure an FTP or FTPS port (see [“Adding an FTP Port” on page 179](#) or [“Adding an FTPS Port” on page 175](#)), you can use the **Passive Mode Listen Address** field to specify the passive mode address for an individual FTP or FTPS port. That way, you can specify a different passive mode address for each FTP port. If an address is specified in the **Passive Mode Listen Address** field and in the `watt.net.ftpPassiveLocalAddr` property, the PORT command uses the value specified in the `watt.net.ftpPassiveLocalAddr` property.

watt.net.ftpPassivePort.max

Specifies the maximum port number of a port range for FTP/FTPS listeners to use with a client data connection that uses passive transfer mode (PASV). Must be used with `watt.ftpPassivePort.min`. For usage information, see `watt.ftpPassivePort.min`.

watt.net.ftpPassivePort.min

Specifies the minimum port number of a port range for FTP/FTPS listeners to use with a client data connection that uses passive transfer mode (PASV). Must be used with `watt.ftpPassivePort.max`. When a port range is specified with these properties, only the ports within the specified minimum and maximum port range (inclusive) are used as the listening ports for incoming FTP/FTPS client data connections. This enables a firewall administrator to open only the specified ports.

Operational considerations:

- If both properties are not present or undefined, FTP/FTPS listeners continue the previous behavior of listening on any free port.
- If the value specified for `watt.net.ftpPassivePort.min` is less than 1, a default value of 1 is used. If the value specified for `watt.net.ftpPassivePort.max` is greater than 65534, a default value of 65534 is used. When both of these conditions exist simultaneously, FTP/FTPS listeners continue the previous behavior of listening on any free port.
- An error message is returned to the FTP/FTPS client on the command channel when the specified values do not fall within the expected range. For example, if one of the properties is not defined, if the `watt.net.ftpPassivePort.min` value is larger than the `watt.net.ftpPassivePort.max` value, or if one of the properties is not a valid number.
- An error message is also returned when all the ports in the specified port range are in use.
- Specific details of the error messages are available in the `serverYYYYMMDD.log` file.

Restarting the Integration Server is not required after defining these settings. You can modify the port range properties in the Integration Server Administrator at any time.

watt.net.ftpSweepInterval

Specifies the frequency, measured in seconds, at which an FTP sweeper executes. The FTP sweeper iterates through the FTP sessions in memory and removes the sessions that have exceeded their allotted idle timeout. By default, the FTP sweeper executes every 600 seconds (10 minutes).

watt.net.ftpUseCertMap

Specifies whether the Integration Server will honor certificate maps for requests received by FTPS ports.

When this property is set to `false` (the default), the Integration Server ignores the user specified on a client certificate and logs the user in with the information provided on the `userid/password` prompt instead.

When this property is set to `true`, if the client certificate has been previously mapped to an Integration Server user, the Integration Server will log the user in as the `userid` specified in the client certificate. The Integration Server ignores the `userid` provided on the `userid/password` prompt.

For example, suppose `watt.net.ftpUseCertMap` is set to `false`, and a certificate has been previously mapped to user Alice. When a user provides a certificate for user Alice and enters Alice's user name and password in response to the prompt, the Integration Server will log the user in as Alice. However, if the user provides the same certificate, but provides Bob's user name and password in response to the prompt, the Integration Server will log the user in as Bob. In other words, the Integration Server ignores the certificate map.

Note:

The **None**, **Request Certificate**, and **Require Certificate** client authentication settings on the FTPS Listener Configuration screen control whether the Integration Server asks for a certificate and how the Integration Server behaves when it does *not* receive one. The `watt.net.ftpUseCertMap` property controls how the Integration Server behaves when it *does* receive a certificate from an FTP client. For more information about client authentication at FTPS and HTTPS ports, see [“Client Certificates” on page 518](#). For more information about certificate mapping, see [“Importing a Certificate \(Client or CA Signing Certificate\) and Mapping It to a User” on page 519](#).

watt.net.ftpUseDefaultContentHdlr

Specifies how the FTP listening port on Integration Server should handle an incoming request with an unrecognized file extension. When set to `true`, the FTP listening port processes the incoming request using the default content handler, which treats the content as `text/html`. When set to `false`, the FTP listening port returns an exception when an incoming request with an unrecognized file extension is received. The default is `true`.

watt.net.httpChunkSize

Sets the default chunk size when sending an HTTP request or response using `Transfer-Encoding:Chunked`. The default chunk size is 8192 bytes. The minimum chunk size is 500 bytes.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.httpPass

The default HTTP password that Integration Server must use when invoking a service as a client.

watt.net.httpUser

Specifies the default authenticated HTTP user name that Integration Server will use while acting as a client to invoke a service. For example, if Integration Server is invoking the `pub.client:http` service without specifying the `auth/user` parameter, then Integration Server uses the value of this property as the user name. There is no default value.

watt.net.http401.retryWithoutCookie

Controls whether Integration Server retries HTTP requests without the cookie header. Set the parameter to `false` to prevent the `pub.client:http` service from retrying a request without a cookie header when Integration Server receives HTTP 401 as a response. The default is `true`.

Note:

The `watt.net.http401.retryWithoutCookie` parameter is introduced for PIE-76895 in IS_10.5_Core_Fix20.

watt.net.http401.throwException

Specifies whether the `pub.client:http` service throws a `NetException` when receiving a 401 error response or, instead, places the HTTP response *header* and *body* in the pipeline. When `watt.net.http401.throwException` is set to `true`, when the `pub.client:http` service receives a 401 error, the service throws a `NetException`. When `watt.net.http401.throwException` is set to `false`, when the `pub.client:http` service receives a 401 error, the service suppresses the `NetException` and places the HTTP response header and body, if one exists, into the *header* and *body* fields in the service output. The default is `true`.

watt.net.http501-599.throwException

Specifies whether the `pub.client:http` service throws a `ServiceException` or returns response headers and response body when receiving a 501 to 599 level response from a remote HTTP server. When set to `true`, the `pub.client:http` service throws a `ServiceException` when it receives a 501 to 599 level response from a remote HTTP server. When set to `false`, the `pub.client:http` service does not throw a `ServiceException` when it receives a 501 to 599 level response from a remote HTTP server. Instead, when the `pub.client:http` service returns a status code in the 501 to 599 range, the service returns the status code, response headers, and response body in the service output. The default is `true`.

Note:

When the remote HTTP server returns a response code of 500, the `pub.client:http` service returns the status code, response headers, and response body.

watt.net.jsse.client.enabledCipherSuiteList

Specifies, using a comma-separated list or a file, the cipher suites used on JSSE sockets that are used while making outbound HTTPS or FTPS requests. To include all the cipher suites supported by the JVM, set this parameter to `default`.

For example:

```
watt.net.jsse.client.enabledCipherSuiteList= TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE
```

```
watt.net.jsse.client.enabledCipherSuiteList=default
```

The default values is `default`.

You can set the value of this parameter to a comma-separated list, default, or the absolute path to a file. You cannot specify a combination of these. For information about specifying a file as the value for this parameter, see [“Specifying Cipher Suites for Use with SSL” on page 466](#).

Note:

The value of the `watt.net.jsse.client.enabledCipherSuiteList` property affects all HTTPS, FTPS, and E-mail ports; and SMTP outbound connections of Integration Server that use JSSE for SSL.

Note:

Any changes you make to `watt.net.jsse.client.enabledCipherSuiteList` affect new connections only.

watt.net.jsse.client.enabledProtocols

Specifies the SSL and TLS protocol versions that Integration Server supports when acting as a client making outbound requests. Specify a comma-separated list that includes one or more of the following:

- SSLv2Hello
- SSLv3
- TLSv1
- TLSv1.1
- TLSv1.2

The default value is: `TLSv1,TLSv1.1,TLSv1.2`

Note:

The value of the `watt.net.jsse.client.enabledProtocols` property affects all HTTPS, FTPS, and E-mail ports; and SMTP outbound connections of Integration Server that use JSSE for SSL.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.jsse.server.enabledCipherSuiteList

Specifies, using a comma-separated list or file, the cipher suites used on Integration Server ports that use JSSE and handle inbound requests.

To include all the cipher suites supported by the JVM, set this parameter to `default`.

For example:

```
watt.net.jsse.server.enabledCipherSuiteList= TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_RSA_W
```

```
watt.net.jsse.server.enabledCipherSuiteList=default
```

The default value is `default`.

You can set the value of this parameter to a comma-separated list, default, or the absolute path to a file. You cannot specify a combination of these. For information about specifying a file as the value for this parameter, see [“Specifying Cipher Suites for Use with SSL” on page 466](#).

Important:

For changes to this property to take effect, you must start the port. If the port is already started, you can restart it by disabling the port and then enabling it.

watt.net.jsse.server.enabledProtocols

Specifies the SSL protocol versions that Integration Server supports when acting as a server handling inbound requests. Specify a comma-separated list that includes one or more of the following:

- SSLv2Hello
- SSLv3
- TLSv1
- TLSv1.1
- TLSv1.2

The default value is: SSLv2Hello,TLSv1,TLSv1.1,TLSv1.2

Note:

These values are case-sensitive. Specify the values exactly as shown. The value of `watt.net.jsse.server.enabledProtocols` affects all HTTPS and FTPS ports that use JSSE for SSL. An HTTPS or FTPS port uses JSSE when the **Use JSSE** parameter for the port is set to **Yes**.

Note:

To prevent a protocol downgrade during negotiation, set `watt.net.jsse.server.enabledProtocols` to a single protocol version that is TLSv1 or higher.

You can also configure the allowed protocols for use with a particular JSSE port by specifying the allowed protocols in the HTTPS and FTPS port records of the `listeners.cnf` file. For more information, see [“Configuring the Allowed Protocols for JSSE per Port” on page 209](#).

Note:

The `jsseEnabledProtocols` value specified for the port record in the `listeners.cnf` file overrides the value set by `watt.net.jsse.server.enabledProtocols`.

watt.net.jsse.server.SSLSessionTimeout

Specifies the amount of time in seconds for which Integration Server waits before timing out and removing an SSL session from the SSL cache. The value of `watt.net.jsse.server.SSLSessionTimeout` must be greater than or equal to 0. If the value is set to 0, then the SSL session will never time out. By default this parameter is empty, which indicates that Integration Server follows the default caching behavior that is supported by JSSE.

Note:

This server configuration parameter is only applicable for ports using JSSE.

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.jsse.server.useCipherSuitesOrder

Specifies whether the local cipher suites preference should be honored by Integration Server during the SSL/TLS handshake when Integration Server acts as the SSL/TLS server and uses JSSE. When set to true, Integration Server uses the cipher suites set in `watt.net.jsse.server.enabledCipherSuiteList` during the SSL/TLS handshake. If `watt.net.jsse.server.enabledCipherSuiteList` is set to "default" or is empty, then Integration Server ignores this parameter. When set to false, Integration Server ignores the order of cipher suites. The default value of this parameter is false.

Important:

For changes to this property to take effect, you must start the port. If the port is already started, you can restart it by disabling the port and then enabling it.

watt.net.localhost

Sets the host name of the machine from which you are using the Integration Server. There is no default.

In some instances, such as when the Integration Server needs to identify itself to other Integration Servers, the IP address of the machine hosting Integration Server is required. In this case, Integration Server performs a reverse DNS lookup on the specified host name and supplies the IP address of the machine in place of the loopback address (127.0.0.1 in IPv4 or ::1 in IPv6), which is sometimes returned by `java.net.InetAddress.getLocalHost()` in place of the actual IP address. In many cases, the loopback address is not sufficient and Integration Server needs the actual address. This most commonly occurs when the IP address of the host is acquired dynamically from a DHCP server, or when the host has more than one network interface card.

In most cases, you can resolve the IP address by modifying the `C:\Windows\system32\drivers\etc` file (in Windows) or the `etc/hosts` or the `etc/nsswitch.conf` files (in Linux and Unix). When you cannot modify these files, or if modifying them does not correct the problem, set `watt.net.localhost`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.maxClientKeepaliveConns

Sets the default number of client keep alive connections to retain for a given target endpoint. The default is 0, which indicates that Integration Server does not retain client keep alive connections for a target endpoint. Integration Server creates a new socket for each request.

Software AG recommends setting `watt.net.maxClientKeepaliveConns` to 0. Setting the property to a value higher than 0 may be beneficial in situations where the frequency and number of concurrent requests to a given target endpoint are high. In situations where this is not the case, idle sockets will become stale and inoperable, resulting in unexpected exceptions such as the following:

```
[ISC.0077.9998E] Exception --> org.apache.axis2.AxisFault: Broken pipe
```

watt.net.maxRedirects

Specifies the maximum number of HTTP redirects to allow before throwing an I/O exception. The default is 5.

watt.net.maxRetries

Specifies the maximum number of retry attempts Integration Server can make for a failed socket connection. The default is 1. A value of 0 indicates that Integration Server should not retry a failed socket connection.

watt.net.overrideSystemProxyselector

Specifies whether the proxy selector of Integration Server will override the default JVM system proxy selector when a Java service tries to connect to a remote server. When this property is set to true, all network connections will honor the proxy aliases configured using Integration Server Administrator. When this property is set to false, the default JVM system proxy selector will be used. The default is false.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.primaryListener

This is an internal property. Do not modify.

watt.net.proxySkipList

Specifies a comma-separated list of domain names for which the Integration Server should not use proxy servers. The default is localhost.

watt.net.proxy.fallbackToDirectConnection

Specifies whether Integration Server should route HTTP, HTTPS, FTP, SFTP, and SOCKS requests directly to the target server when connections through all of the specified proxy server aliases for the requested protocol fail. For example, if the request uses HTTP, Integration Server routes the request through an HTTP proxy server alias. If this property is set to false and the connection to the destination server through proxy aliases fails, Integration Server issues an exception. If this property is set to true, Integration Server attempts to make a direct connection with the destination server specified in the request. The default is true.

Note:

If there are no proxy server aliases defined for Integration Server, the value of `watt.net.proxy.fallbackToDirectConnection` is ignored. For information about proxy server aliases, see [“Specifying a Default Proxy Server Alias” on page 125](#).

watt.net.proxy.useNonDefaultProxies

Specifies whether Integration Server makes outbound connection requests using all enabled proxy server aliases if the outbound request does not specify a proxy server alias and if a default proxy server alias is not specified. When the `watt.net.proxy.useNonDefaultProxies` parameter is set to true, if the outbound request does not specify a proxy server alias and there is no default proxy server alias, Integration Server makes outbound requests using each enabled proxy server alias until the request is sent successfully or all proxy servers have been tried. If all proxy servers have been tried and the attempt to send the request fails or if there are no proxy aliases specified, Integration Server either makes a direct connection to the target server or throws an exception depending on the settings specified for the `watt.net.proxy.fallbackToDirectConnection` parameter. When the `watt.net.proxy.useNonDefaultProxies` parameter is set to false, if a default proxy server alias is not specified, Integration Server sends the request to the remote server using a direct connection. Integration Server does not attempt to make outbound requests using the enabled proxy server aliases. The default is true.

For more information about how Integration Server uses proxy servers, see [“How Integration Server Uses Proxy Servers” on page 119](#).

watt.net.retries

Specifies the number of times to retry a server that times out. This can be overridden by the client. The default is 0.

watt.net.sftpSweepInterval

Specifies the frequency, measured in minutes, at which an SFTP sweeper executes. The SFTP sweeper iterates through the SFTP sessions in memory and removes the sessions that have exceeded their allotted idle timeout. By default, the SFTP sweeper executes every 10 minutes.

watt.net.socketpool.sweeperInterval

Specifies the frequency, in seconds, at which the socket pool sweeper executes. The socket pool sweeper sends a ping request to all webMethods Enterprise Gateway connections and HTTP client connections. During a sweep it removes any invalid HTTP client connections. By default, the sweeper executes every 60 seconds.

Note:

The value of `watt.net.socketpool.sweeperInterval` should be less than the value of the `watt.server.rg.internalregistration.timeout` server configuration parameter.

Note:

On your Enterprise Gateway, if `watt.server.rg.gateway.pinginterval` is set, Integration Server uses that setting and ignores the value of `watt.net.socketpool.sweeperInterval`.

watt.net.socketProvider

Identifies the Java class that implements the `com.wm.net.SocketProviderIf` interface for secure socket communication. The default is `com.wm.ext.iaik.IaikSecureSocket`.

watt.net.ssl.client.cipherSuiteList

Specifies a list of cipher suites for outbound SSL connections. When the default value is set to `default`, Integration Server uses its default list of cipher suites. If you want to specify non-default cipher suites, enter a comma-separated list of cipher suite names. If the property `watt.net.ssl.client.strongcipheronly` is set to `true`, and if there are any non-strong cipher suites in the list specified, those will be ignored, and a warning message will be logged.

You can set the value of this parameter to a comma-separated list, `default`, or the absolute path to a file. You cannot specify a combination of these. For information about specifying a file as the value for this parameter, see [“Specifying Cipher Suites for Use with SSL” on page 466](#).

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.client.handshake.maxVersion

Specifies the maximum SSL protocol version that Integration Server supports when Integration Server acts as a client and makes an outbound request. For example, if set to `tls` (the default), the maximum version of SSL protocol supported by Integration Server is TLS 1.0. If set to `sslv3`, the maximum version of SSL protocol supported by Integration Server is SSL 3.0.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.client.handshake.minVersion

Specifies the minimum version of SSL protocol Integration Server supports when Integration Server acts as a client and makes an outbound request. Set to:

- `sslv2` (the default) to specify SSL 2.0
- `tls` to specify TLS 1.0
- `sslv3` to specify SSL 3.0

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.client.hostnameverification

When Integration Server is acting as an HTTPS client, this parameter specifies whether Integration Server should restrict outbound HTTPS connections only when a valid hostname is found in the server's certificate.

- When set to `true`, Integration Server verifies if the hostname is present in the server's certificate. If this verification fails, an error is logged and the connection is aborted.
- When set to `false`, Integration Server will bypass the hostname verification. This is the default.
- When set to `log`, Integration Server logs the debug message in the server log if the hostname verification fails, but allows the connection to go through. If the hostname verification succeeds, no log is generated.

watt.net.ssl.client.strongcipheronly

Specifies whether the Integration Server is to restrict outbound HTTPS connections to use strong cipher suites only (128 bit session keys or higher). If you specify `false` (the default), when Integration Server initiates a connection to another server, it will attempt to negotiate a strong cipher suite, and if unsuccessful will fall back to using a weak (64, 56, or 40 bit) cipher suite. If you specify `true`, when Integration Server initiates a connection to another server, it will attempt to negotiate a strong cipher suite, and if unsuccessful will disconnect rather than use a weak cipher suite.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.client.ftps.useJSSE

Controls the use of JSSE for all of the outbound FTPS connections from Integration Server. Set this parameter to `true` to use JSSE for all of the outbound FTPS connections. Set this property to `false` to indicate that JSSE is not used for outbound FTPS connections. The default is `false`.

Note:

When executing the `pub.client.ftp` service or the `pub.client.ftp.login` service, the value of the `useJSSE` input parameter overrides the value of the `watt.net.ssl.client.ftps.useJSSE` server configuration parameter.

watt.net.ssl.client.useJSSE

Controls the use of JSSE for all of the outbound HTTPS connections from Integration Server. Set this parameter to `true` to use JSSE for all of the outbound HTTPS connections. Set this property to `false` to indicate that JSSE is not used for outbound HTTPS connections. The default is `true`.

Note:

When executing the `pub.client.http` service or the `pub.client.soapClient` service, the value of the `useJSSE` input parameter overrides the value of the `watt.net.ssl.client.useJSSE` server configuration parameter.

watt.net.ssl.email.client.useJSSE

Controls the use of JSSE for all the outbound SMTP connections from Integration Server. Set this parameter to `true` to use JSSE for all of the outbound SMTP connections. Otherwise, set it to `false`. The default value is `true`.

Note:

When executing the `pub.client.smtp` service, the value of the `useJSSE` input parameter of the service is used instead of this property.

watt.net.ssl.randomAlgorithm

Identifies the random algorithm name used by Integration Server. The default value is `FIPS186_2usingSHA1`.

Note:

This parameter is for use only when Integration Server is installed on HP-UX.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.cipherSuiteList

Specifies a list of cipher suites for inbound SSL connections. When the default value is set to `default`, Integration Server uses its default list of cipher suites. If you want to specify non-default cipher suites, enter a comma-separated list of cipher suite names. If the property `watt.net.ssl.server.strongcipheronly` is set to `true`, and if there are any non-strong cipher suites in the list, those will be ignored and a warning message will be logged.

You can set the value of this parameter to a comma-separated list, `default`, or the absolute path to a file. You cannot specify a combination of these. For information about specifying a file as the value for this parameter, see [“Specifying Cipher Suites for Use with SSL” on page 466](#).

Important:

If you change the setting of this parameter, you must restart any impacted ports for the changes to take effect. To restart a port, you can disable and then enable the port. Reloading the package associated with the port or restarting Integration Server also restarts a port.

watt.net.ssl.server.clientHandshakeTimeout

Specifies the number of milliseconds that Integration Server waits for a response during an SSL handshake before timing out. Integration Server uses this value for inbound and outbound requests. The default is 20000 milliseconds.

watt.net.ssl.server.handshake.maxVersion

Specifies the maximum version of the SSL protocol that Integration Server supports when acting as the server handling inbound requests. Set to:

- `sslv3` to specify SSL 3.0
- `tls` to specify TLS 1.0

The default is `tls`.

The value of `watt.net.ssl.server.handshake.maxVersion` affects all HTTPS ports that use Entrust toolkit for SSL. An HTTPS port uses Entrust when the **Use JSSE** parameter for the port is set to **No**.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.handshake.minVersion

Specifies the minimum version of the SSL protocol that Integration Server supports when acting as the server handling inbound requests. Set to:

- `sslv3` to specify SSL 3.0
- `tls` to specify TLS 1.0.

The default is `tls`.

The value of `watt.net.ssl.server.handshake.minVersion` affects all HTTPS ports that use Entrust toolkit for SSL. An HTTPS port uses Entrust when the **Use JSSE parameter** for the port is set to **No**.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.sessionlog

Specifies whether Integration Server logs the SSL session information for inbound connections. When this parameter is set to `true`, Integration Server logs the SSL session in a separate `inboundSSLSessions.log` file for Entrust and JSSE security providers. Integration Server creates the `inboundSSLSessions.log` file under *Integration Server_directory* \instances*instance_name*\logs directory. SSL session information is logged in JSON format. When this parameter is set to `false`, Integration Server does not log the SSL session information. The default value is `false`.

watt.net.ssl.server.sessionlog.maxFileSize

Specifies the maximum size of the `inboundSSLSessions.log` file in megabytes (MB). When the file reaches the maximum size, Integration Server renames the file to `inboundSSLSessions_<DATE(YYYYMMDD)>_TIME(HHMMSS).log` and creates a new

inboundSSLSessions.log file. Specify an integer greater than 0 (zero). If you specify a value less than or equal to zero, then Integration Server uses the default value. The default value is 10.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.sessionlog.cacheLogEntries

Specifies whether Integration Server tracks the SSL session log entries in cache. If this parameter is set to `true`, Integration Server does not log the SSL session information for entries that already exist in the cache. This eliminates duplicate entries in the log file. Default value is `false`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime

Specifies, in seconds, how often Integration Server checks for and removes the expired SSL session log entries from cache. If a client reuses a session for which Integration Server has removed the log entry, then Integration Server logs the session information again for that session. Specify an integer greater than 0 (zero). If you specify a value less than or equal to zero, then Integration Server uses the default value. The default is 300 seconds.

Integration Server uses a sweeper task called SSL Session Log Entries Sweeper to remove the expired sessions.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.sessionlog.file

Specifies either an absolute or relative path to the file to which Integration Server writes the SSL session information. Relative path is relative to the Integration Server home directory: *Integration Server_directory \instances\instance_name*. You must specify a path with a valid directory name and filename. The default is: *Integration Server_directory \instances\instance_name\logs\inboundSSLSessions.log*.

Note:

If you do not provide a valid path, then Integration Server uses the default path and file name: *Integration Server_directory \instances\instance_name\logs\inboundSSLSessions.log*.

If you specify a path that points to a network location, when the location is inaccessible to log the session details, Integration Server logs the SSL session information to the console till the location becomes accessible. Additionally, if Integration Server faces a connection problem while logging the session details, then the log file may contain null or invalid characters.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.sessionlog.prettyPrint

Specifies whether the SSL session log entry is formatted with carriage returns and indentation to make the SSL session log easier to read. If this parameter is set to true, Integration Server formats JSON with carriage returns and indentation to ease readability. The default is false.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.sessionlog.includeTimestamp

Specifies whether Integration Server includes a timestamp in the SSL session log entries. If this parameter is set to true, log entry begins with a timestamp. The default is false.

Example:

If this parameter is set to true, then the log entry starts with a timestamp:2019-07-03 10:37:24 IST {"provider":"JSSE","loggedInUser":"Administrator"...}. If this parameter is set to false, then the log entry does not contain a timestamp:{"provider":"JSSE","loggedInUser":"Administrator"...}

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.net.ssl.server.strongcipheronly

Specifies whether the Integration Server is to restrict inbound HTTPS connections to use strong cipher suites only (128 bit session keys or higher). If you specify false (the default), when a client connects to the Integration Server, the server will attempt to negotiate a strong cipher suite, and if unsuccessful will fall back to using a weak (64, 56, or 40 bit) cipher suite. If you specify true, when a client connects to the Integration Server, the server will attempt to negotiate a strong cipher suite, and if unsuccessful will disconnect rather than use a weak cipher suite.

Important:

If you change the setting of this parameter, you must restart any impacted ports for the changes to take effect. To restart a port, you can disable and then enable the port. Reloading the package associated with the port or restarting Integration Server also restarts a port.

watt.net.timeout

Specifies the number of seconds the server waits for an HTTP request to be fulfilled before the request times out. To set Integration Server to wait indefinitely for a response from the target server, set this parameter to 0. The default is 300 (5 minutes).

Important:

If you set watt.net.timeout to 0 and the target server does not respond to the request, the Integration Server making the request cannot process new requests due to thread pool exhaustion.

watt.net.useCookies

Specifies whether Integration Server accepts or denies cookies when communicating with web server. Set to true to accept cookies; set to false (or null) to deny cookies. The default is true.

watt.net.userAgent

Specifies the value the server uses in the HTTP User Agent request header when it requests a web document from a web server. The default is `Mozilla/4.0 [en] (WinNT; I)`.

watt.security.

watt.security.cert.wmChainVerifier.enforceExtensionsChecks

Specifies whether Integration Server is to validate (`true`) or not validate (`false`) certificate extensions, if any, when the server performs certificate verification. The default is `false`.

watt.security.cert.wmChainVerifier.trustByDefault

In cases where no directory or a directory containing no certificates is specified for the Trusted Certificates directory, specifies whether the server is to trust:

- Certificates presented by peer servers (in response to this server's outbound request)
- S/MIME signatures

Specifies whether the server is to trust (`true`) or not trust (`false`) certificates and S/MIME signatures in this situation. The default is `true`. For improved security, Software AG recommends that you set this parameter to `false` and specify a Trusted Certificates directory.

watt.security.decrypt.keyAlias

Specifies the alias of the default private key Integration Server uses for decryption.

watt.security.decrypt.keyStoreAlias

Specifies the alias of the keystore containing the default private key Integration Server uses for decryption.

watt.security.fips.mode

Specifies whether the server is to support FIPS (Federal Information Processing Standards). The default is `false`. If this parameter is set to `true`, the server initializes FIPS as part of server startup. If FIPS initialization fails, the error is logged to `server.log` and the server shuts down.

watt.security.kerberos.client.useSPNEGO

Specifies whether Integration Server generates a SPNEGO-based Kerberos ticket for all outbound requests that use Kerberos authentication. When set to `true`, for all outbound requests that use Kerberos authentication, Integration Server generates a SPNEGO token using SPNEGO OID (Object Identifier). When set to `false`, Integration Server generates a Java Kerberos ticket using the JGSS Kerberos OID. The default value for this property is `false`.

watt.security.KeystoreAndTruststore.defaultAliasCreated

This is an internal parameter. Do not modify.

watt.security.keyStore.supportedTypes

Specifies the keystore types supported by Integration Server. Each supported type is separated by a comma. The default values are `JKS` and `PKCS12`.

To specify additional keystore types for Integration Server, you need to do the following:

1. Add the new keystore type to the list of values for this property.
2. Add the keystore provider, using one of the following methods:
 - a. Using the "Add Security Provider" link in Integration Server Administrator.

- b. Modifying the "java.security" file of the JVM (you must use this method for a PKCS11-type keystore).

Note: Software AG does not guarantee the behavior of additional keystore types, and does not provide support for them.

watt.security.max.contentLength

Specifies the maximum size limit of the payload for an HTTP request. If `watt.security.max.contentLength` is greater than 0 and Integration Server receives an HTTP request with a Content-Length header greater than `watt.security.max.contentLength`, Integration Server rejects the request with a 413 Payload Too Large error and then closes the connection. If `watt.security.max.contentLength` is greater than 0 and Integration Server receives a request with no Content-Length in the request header, Integration Server rejects the request with a 411 Length Required error and then closes the connection. Set this parameter to a value greater than 0 to enforce a maximum Content-Length. The default value for `watt.security.max.contentLength` is 0 which means that Integration Server does not enforce a maximum Content-Length.

Note:

Take care when setting the `watt.security.max*` server configuration parameters. If the parameters values are set too low, Integration Server rejects legitimate requests. It is also possible to lock yourself out of Integration Server Administrator by setting the parameters too low. If this happens, you can try changing the configuration parameter values by invoking the `wm.server.admin:setSettings` service with a command-line HTTP client. If you are still unable to change the value of these parameters, you need to kill the Integration Server process, manually edit the parameters in the *Integration Server_directory/instances/instanceName/config/server.cnf* file, and restart Integration Server.

watt.security.max.headerLength

Specifies the maximum length of the header for an HTTP request. If `watt.security.max.headerLength` is greater than 0 and a request received by Integration Server has a header longer than `watt.security.max.headerLength`, Integration Server rejects the request with a 431 Request Header Fields Too Large error and then closes the connection. When determining the length of the header, Integration Server considers the keys and values of all the request header fields and all the whitespace characters in the request header. Set this parameter to a value greater than 0 to enforce a maximum request header length. The default value for `watt.security.max.headerLength` is 0 which means that Integration Server does not enforce a maximum header length.

Note:

Take care when setting the `watt.security.max*` server configuration parameters. If the parameters values are set too low, Integration Server rejects legitimate requests. It is also possible to lock yourself out of Integration Server Administrator by setting the parameters too low. If this happens, you can try changing the configuration parameter values by invoking the `wm.server.admin:setSettings` service with a command-line HTTP client. If you are still unable to change the value of these parameters, you need to kill the Integration Server process, manually edit the parameters in the *Integration Server_directory/instances/instanceName/config/server.cnf* file, and restart Integration Server.

watt.security.max.urlLength

Specifies the maximum length of the URL for an HTTP request. If `watt.security.max.urlLength` is greater than 0 and Integration Server receives a request with a URL longer than `watt.security.max.urlLength`, Integration Server rejects the request with a 414 URI Too Long error and then closes the connection. When determining the length of the URL, Integration Server considers the path and the query parts of the URL. Set this parameter to a value greater than 0 to enforce a maximum URL length. The default value for `watt.security.max.urlLength` is 0 which means that Integration Server does not enforce a maximum URL length.

Note:

Take care when setting the `watt.security.max*` server configuration parameters. If the parameters values are set too low, Integration Server rejects legitimate requests. It is also possible to lock yourself out of Integration Server Administrator by setting the parameters too low. If this happens, you can try changing the configuration parameter values by invoking the `wm.server.admin:setSettings` service with a command-line HTTP client. If you are still unable to change the value of these parameters, you need to kill the Integration Server process, manually edit the parameters in the *Integration Server_directory/instances/instanceName/config/server.cnf* file, and restart Integration Server.

watt.security.ope.AllowInternalPasswordAccess

Specifies whether the built-in services supporting OPE (outbound password encryption) for flow services may access the Integration Server's internal passwords. If this parameter is set to `true`, the OPE services may access the internal passwords. If it is set to `false`, the OPE services are not allowed access to the internal passwords. By default, this parameter is set to `false`.

Internal passwords are passwords for use by the Integration Server itself to access secure resources (e.g., remote Integration Servers, JDBC connection pools, LDAP servers, etc.). Internal passwords are managed using the Integration Server Administrator and are stored in the outbound password store. Flow services are also allowed to store passwords in the outbound password store. However, by default, passwords stored by a flow service are considered public, as opposed to `internal`. This distinction allows flow services to use the outbound password store as a secure mechanism for storing and retrieving passwords, but protects the Integration Server's internal passwords.

You can allow flow services to access internal passwords (i.e., store, retrieve, and modify) by setting `watt.security.ope.AllowInternalPasswordAccess` to `true`. However, this should be done only if you explicitly wish to have a flow service work with internal passwords. Otherwise, it is recommended you deny access to internal passwords by setting `watt.security.ope.AllowInternalPasswordAccess` to `false`.

watt.security.openid.logExceptions

Specifies whether Integration Server writes OpenID errors to the error log. When the OpenID authentication process encounters errors, Integration Server returns `error` and `error_description` variables in the body of the HTTP response. When `watt.security.openid.logExceptions` is `true`, the default, Integration Server throws an exception from the redirection endpoint service, causing the error to be written to the error log. To stop Integration Server from writing OpenID errors to the error log, set `watt.security.openid.logExceptions` to `false`.

watt.security.pub.getFile.checkReadAllowed

Specifies whether the `pub.file.getFile` service is to check the `allowedReadPaths` property in the `fileAccessControl.cnf` file to determine if the requested file can be retrieved from the file system. The `allowedReadPaths` property contains a list of directories to which the services in the `pub.file` folder have read permission.

When `watt.security.pub.getFile.checkReadAllowed` is set to `true`, the `pub.file:getFile` service checks the `allowedReadPaths` property in the `fileAccessControl.cnf` file. If the requested file or file directory is listed, the `pub.file:getFile` service returns its contents to the pipeline.

If `watt.security.pub.getFile.checkReadAllowed` is set to `true` and the file or file directory is not listed in the `allowedReadPaths` property, the `pub.file:getFile` service throws an exception.

When `watt.security.pub.getFile.checkReadAllowed` is set to `false`, the `pub.file:getFile` service retrieves the specified file from the local file system without checking the `fileAccessControl.cnf` file.

Changes to this property take effect immediately. However, if you modify the `fileAccessControl.cnf` file, the `WmPublic` package must be reloaded before the changes take effect.

For more information about the `pub.file:getFile` service and configuring the `fileAccessControl.cnf` file, see *webMethods Integration Server Built-In Services Reference*.

watt.security.session.forceReauthOnExpiration

Specifies whether Integration Server accepts or rejects a request that includes an expired or invalid session. When set to `true`, Integration Server rejects any request that includes a cookie identifying an expired or invalid session, even if the request includes valid user credentials. The rejection response directs the browser to clear its session identifier and to prompt the user for credentials. When set to `false`, Integration Server creates a new session using the credentials in the cookie. The default value is `true`. A value of `true` offers more secure behavior.

watt.security.sign.keyAlias

Specifies the alias of the default private key Integration Server uses to digitally sign documents.

watt.security.sign.keyStoreAlias

Specifies the alias of the keystore containing the default private key Integration Server uses to digitally sign documents.

watt.security.ssl.cacheClientSessions

Controls whether Integration Server reuses previous SSL session information (for example, client certificates) for connections to the same client. When this property is set to `true`, Integration Server caches and reuses SSL session information. For example, set this property to `true` when there are repeated HTTPS requests from the same client. If set to `false` (the default), Integration Server does not cache the sessions and creates a new session for every SSL handshake.

Note:

Setting the property to `false` might decrease performance.

watt.security.ssl.cachedClientSessions.sweeperInterval

Specifies, in milliseconds, how often Integration Server checks for and removes expired SSL client sessions from its session cache. The default value is 600000 milliseconds (10 minutes).

watt.security.ssl.client.ignoreEmptyAuthoritiesList

Specifies whether an Integration Server acting as a client sends its certificate chain after a remote SSL server returns an empty list of trusted authorities. When set to `true`, Integration Server disregards the empty trusted authorities list and sends its chain anyway. When set to `false`, before sending out its certificate chain, Integration Server requires the presentation of trusted authorities list that proves itself trusted. The default is `false`.

watt.security.ssl.ignoreExpiredChains

Specifies whether the Integration Server ignores expired CA certificates in a certificate chain it receives from an Internet resource (i.e., a web server, another Integration Server). To have the Integration Server ignore expired CA certificates and allow SSL connections when a certificate is expired, set the `watt.security.ssl.ignoreExpiredChains` setting to `true`. Note that this is less secure than denying connections when a certificate is expired. The default is `false`. For more information about this setting, see “[Integration Server as an SSL Server](#)” on page 459.

watt.security.ssl.keyAlias

Specifies the alias of the Integration Server default SSL private key.

watt.security.ssl.keypurposeverification

When Integration Server is acting as an HTTPS client, this parameter specifies whether the server should restrict outbound HTTPS connections only when a valid Extended Key Purpose field is present in the server's certificate. The content of the Key Purpose field, `id-kp-serverAuth`, should be in the IETF-mandated format, `TLS WWW server authentication` for the verification to pass. Refer to the section titled *Extended Key Usage*, in the document <http://www.ietf.org/rfc/rfc3280.txt> for more information regarding this format.

Three values are allowed for this `watt` property - `true`, `false` and `log`.

- When set to `true`, it will verify the presence of the key purpose field in the server's certificate. If the key purpose verification fails, an error is logged and the connection is aborted. If the verification succeeds, no error is logged.
- When set to `false`, it will bypass the verification of the key purpose field. The default is `false`.
- When set to `log`, it will log a debug message in the server log if the key purpose field verification fails. `watt.security.ssl.keyStoreAlias` Specifies the alias of the Integration Server default SSL keystore.

watt.security.ssl.keyStoreAlias

Specifies the alias of the Integration Server default SSL keystore.

watt.security.ssl.resumeClientSessions

Controls whether Integration Server acting as a client, reuses the previous SSL session information for connections to the same server.

- If `watt.security.ssl.resumeClientSessions` is to set to `true` and:
 - Integration Server acts as an HTTPS client, then Integration Server caches the SSL sessions and reuses the SSL session information for future requests to the same server.
 - Integration Server acts as an FTPS client, then Integration Server uses the same SSL session for command and data channel.
- If `watt.security.ssl.resumeClientSessions` is to set to `false` and:
 - Integration Server acts as an HTTPS client, then Integration Server does not cache the SSL sessions and creates a new SSL session for every SSL handshake.
 - Integration Server acts as a FTPS client, then does not cache the SSL sessions and creates a new SSL session for command and data channel. The default value is `false`.

watt.security.trustStoreAlias

Specifies the alias for the truststore that Integration Server uses as the default truststore.

When Integration Server is acting as a server, it uses the default truststore to verify the trust relation when no truststore is provided. For example, Integration Server uses the default truststore when no truststore is provided for HTTPS/FTPS ports or for web service security when there is no truststore at the provider web service endpoints alias.

When Integration Server is acting as a client, most of the components in Integration Server (for example, HTTPS, FTPS, remote server alias) always use the default truststore to verify the trust relation with the server. However, for web service security, Integration Server only uses the default truststore when the user does not provide a truststore in the consumer endpoint alias.

watt.security.trustStore.supportedTypes

Specifies the truststore types supported by Integration Server. Each supported type is separated by a comma. The default value is JKS.

To specify additional truststore types for Integration Server, you need to do the following:

1. Add the new truststore type to the list of values for this property.
2. Add the truststore provider, using one of the following methods:
 - a. Using the "Add Security Provider" link in Integration Server Administrator.
 - b. Modifying the "java.security" file of the JVM.

Note: Software AG does not guarantee the behavior of additional truststore types, and does not provide support for them.

watt.server.

watt.server

This is an internal parameter. Do not modify.

watt.server.acl.groupScanInterval

Specifies how often, in milliseconds, Integration Server checks for the availability of the My webMethods Server database, if the database is unavailable at Integration Server startup. While the database is unavailable, group information about users that are managed by Central User Management is unavailable. As a result, users in those groups do not have the access rights granted by their associated ACLs. While the My webMethods Server database is unavailable, the **Security > Access Control List** screen displays the affected groups with the suffix @--@, for example FinanceGroup@--@. When the My webMethods Server database becomes available, the groups regain their access rights and Integration Server removes the @--@ suffix. The default setting is 10,000 milliseconds (10 seconds).

watt.server.allowDirective

Restricts the use of specified directives to specified ports. For information on directives, see ["Controlling the Use of Directives" on page 499](#)). The syntax for this property is:

port-string is a comma-delimited list of port numbers such as "5555,6666".

Suppose you want to allow all ports to use the default directive, but you want specific ports to use the other directives, as described below:

Restrict use of the invoke directive to ports 5555 and 7777

Restrict use of the web directive to ports 6666 and 7777

Restrict use of the SOAP directive to port 7777

To obtain the behavior described above, you would specify the following:

```
watt.server.allowDirective=invoke,5555,7777,web,6666,7777,soap,7777
```

watt.server.apiportal.url

Specifies the URL for establishing a connection to API Portal and publishing a REST API descriptor. This parameter derives the API Portal URL from the following:

- Host name and port number for the API Portal connection.
- Tenant for which the REST API descriptor is to be published.
- Specification of the REST API descriptor.

watt.server.audit.dbEncoding

Specifies the character set used by the audit logging database. The default is UTF-8. The value for this property must be a standard Internet Assigned Numbers Authority (IANA)-assigned character set name, as defined in the IANA Character Sets specification.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.server.audit.disableInternalServerConnectionSuccessLogging

Set to `true` if you do not want the security log to have the entry "Internal Server successfully authenticated to the Enterprise Gateway." The default value is `false`.

Note:

If this property is `true`, it is applied even when the **Generate Auditing Data on** property on the **Settings > Logging > Edit Security Logger Details** page of IS Administrator is set to "Success" or "Success or Failure".

watt.server.audit.displayLogs.convertTime

Specifies how dates are displayed when you view the service, error, session, guaranteed delivery, and security logs from the Integration Server Administrator. When this property is set to `false`, time stamps are shown as GMT/UTC. When this property is set to `true` (the default), time stamps are shown in the local time zone of the Integration Server, and in the format specified by `watt.server.dateStampFmt`.

watt.server.audit.failFastLoggers

Specifies the loggers that can enter into fail-fast mode when fail-fast mode is enabled for the ISCoreAudit functional alias.

- If `watt.server.audit.failFastLoggers` is empty, all synchronous audit loggers that use a database destination will have fail-fast capability.
- If `watt.server.audit.failFastLoggers` is not empty, only the synchronous audit loggers with a database destination listed in `watt.server.audit.failFastLoggers` will have fail-fast capability.

Note:

Fail-fast mode is enabled for the ISCoreAudit functional alias when the **Fail-Fast Mode Enabled** option is set to **Yes** for the alias.

If you want to identify the specific audit loggers that can enter fail-fast mode, use the `watt.server.audit.failFastLoggers` to specify a comma-separated list of the loggers that can enter into fail-fast mode. For example, if you want the Service Logger and Session Logger to be able to enter into fail-fast mode, set the property as follows:

```
watt.server.audit.failFastLoggers=Service Logger,Session Logger
```

You must specify the word "Logger" as part of the logger name. The property is case-sensitive. You must specify the logger name as it appears on the **Settings > Logging** page of Integration Server Administrator. Do not include a space before or after the comma that separates logger names.

Only synchronous loggers can enter fail-fast mode. Integration Server ignores any asynchronous loggers specified in the `watt.server.audit.failFastLoggers` property.

The default for `watt.server.audit.failFastLoggers` is empty. That is, the default is that all synchronous audit loggers that use a database destination will have fail-fast capability when the ISCoreAudit functional specifies **Yes** for the **Fail-Fast Mode Enabled** option.

watt.server.audit.file.fieldDelimiter

Specifies the character sequence to use to delimit fields within a log record in a file-based audit log. A valid delimiter is any character sequence where a character is any visible (printing) character (0021-007E) from the US ASCII character set. When specifying delimiter character sequences, do not use a character sequence likely to appear in a log entry. To use delimiter characters in file-based audit logs, you must specify a valid value for the `watt.server.audit.file.fieldDelimiter` and `watt.server.audit.file.recordDelimiter` parameters. If a valid value is not specified for both parameters, Integration Server writes log entries using the fixed-length format.

There is no default value for this parameter. By default, Integration Server uses fixed-length field entries for file-based audit logs.

To revert to fixed-length field entries for file-based audit logs, delete the values of the `watt.server.audit.file.fieldDelimiter` and `watt.server.audit.file.recordDelimiter` parameters and restart Integration Server.

After switching between fixed-length format and character-delimited format, Integration Server Administrator cannot display entries from the previous format.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.server.audit.file.recordDelimiter

Specifies the character sequence to use to delimit records in a file-based audit log. A valid delimiter is any character sequence where a character is any visible (printing) character (0021-007E) from the US ASCII character set. When specifying delimiter character sequences, do not use a character sequence likely to appear in a log entry. To use delimiter characters in file-based audit logs, you must specify a valid value for the `watt.server.audit.file.fieldDelimiter` and `watt.server.audit.file.recordDelimiter` parameters. If a valid value is not specified for both parameters, Integration Server writes log entries using the fixed-length format.

There is no default value for this parameter. By default, Integration Server uses fixed-length field entries for file-based audit logs.

To revert to fixed-length field entries for file-based audit logs, delete the values of the `watt.server.audit.file.fieldDelimiter` and `watt.server.audit.file.recordDelimiter` parameters and restart Integration Server.

After switching between fixed-length format and character-delimited format, Integration Server Administrator cannot display entries from the previous format.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.server.audit.logDir

Specifies the directory that contains the audit log files. You can specify the full-path name, or a relative path in relation to the Integration Server. The directory must exist on the server. The default value is `logs`, which indicates the *Integration Server_directory* `\instances\instance_name\logs` directory. If an invalid value is specified, Integration Server uses the default value and writes an error to the server log during startup.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

Note that `watt.server.audit.logDir` does not affect audit loggers that write to a database.

watt.server.audit.logFilesToKeep

Specifies the number of audit log files, including the current log file for the audit logger, that Integration Server keeps on the file system for an audit logger that writes to a file. When Integration Server reaches the limit for the number of log files for the audit logger, each time Integration Server rotates the audit log, Integration Server deletes the oldest archived audit log file. If you set `watt.server.audit.log.filesToKeep` to 1, Integration Server keeps the current audit log file and no previous audit log files for each file-system based audit logger. That is, when Integration Server rotates the audit log for a logger, Integration Server does not create an archive file for the previous audit log. If you set `watt.server.audit.logFilesToKeep` to 0, or any value less than 1, Integration Server keeps an unlimited number of audit log files.

The default value of `watt.server.log.filesToKeep` is 0, indicating that there is no limit to the number of audit log files that Integration Server maintains for audit loggers that write to a file.

The `watt.server.audit.logFilesToKeep` parameter affects only the audit loggers configured to write to a file. The parameter does not affect audit loggers configured to write to a database nor does it affect the `FailedAuditLog`.

If you reduce the number of logs that Integration Server keeps for file-based audit logs and then restart Integration Server, the existing audit logs will not be pruned until Integration Server writes to the audit log. For example, if the error logger writes to a file and you reduce the number of log files to keep from 10 to 6, Integration Server does not delete the 4 oldest error audit log files immediately after start up. Integration Server deletes the 4 oldest error audit logs after the error logger writes to the error audit log.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.audit.logRotateSize

Specifies the file size at which Integration Server rolls over the audit log for a logger that writes to a file. Set this property to N [KB|MB|GB], where N is any valid integer. The minimum size at which Integration Server rotates an audit log is 33KB. If you use KB as the unit of measure, you must set N to a value greater than or equal to 33. If you do not specify a unit of measure, Integration Server treats the supplied N value as bytes. In this case, N must be greater than or equal to 32768 to take effect. Do not include any spaces between the integer and the unit of measure.

There is no default value for this parameter. That is, by default, the parameter has no value. If no value is specified for `watt.server.audit.logRotateSize`, Integration Server rotates the audit logs at midnight only.

If an invalid value is specified, Integration Server proceeds as if no value was specified for the parameter. If you set the value using the **Extended Settings** page in Integration Server Administrator, validation prevents an invalid value from being saved.

The `watt.server.audit.logRotateSize` parameter affects only the audit loggers configured to write to a file. The parameter does not affect audit loggers configured to write to a database.

For more information about audit logging, see the *webMethods Audit Logging Guide*.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.audit.schemaName

Specifies the name of the database schema that Integration Server should use while requesting metadata for the audit logging database. There is no default. If `watt.server.audit.schemaName` is not set, Integration Server does not retrieve database metadata and assumes the lengths of the `WMSERVICECUSTOMFLDS.STRINGVALUE` and `WMSERVICEACTIVITYLOG.FULLMESSAGE` columns are 512 and 1024, respectively.

Note:

Some databases are case-sensitive. When specifying the value for `watt.server.audit.schemaName`, you should match the case of the schema name with the schema name required by the database.

Note:

If you change the value of this property, you must restart Integration Server for the changes to take effect.

watt.server.audit.stdout.fieldDelimiter

Specifies the character string to delimit audit log fields within a log entry when the audit log is written to the console (STDOUT). The default value is `"|"` (i.e. double pipe). The values for `watt.server.audit.stdout.fieldDelimiter` and `watt.server.audit.stdout.recordDelimiter` must be different. When specifying delimiter character sequences, do not use a character sequence likely to appear in a log entry.

You can configure an audit log to write to STDOUT using the environment variable `SAG_IS_AUDIT_STDOUT_LOGGERS`. For more information about environment variables, see [“Environment Variables for Use with Docker” on page 1183](#).

Note:

If you change the value of this property, you must restart Integration Server for the changes to take effect.

watt.server.audit.stdout.recordDelimiter

Specifies the character string to delimit the audit log records when the audit logger is written to the console (STDOUT). The default value is "&&" (i.e. double ampersand). The values for `watt.server.audit.stdout.fieldDelimiter` and `watt.server.audit.stdout.recordDelimiter` must be different. When specifying delimiter character sequences, do not use a character sequence likely to appear in a log entry.

You can configure an audit log to write to STDOUT using the environment variable `SAG_IS_AUDIT_STDOUT_LOGGERS`. For more information about environment variables, see [“Environment Variables for Use with Docker” on page 1183](#).

Note:

If you change the value of this property, you must restart Integration Server for the changes to take effect.

watt.server.audit.um.sessionPool.min

Specifies the minimum number of sessions in the Universal Messaging session pool. Integration Server maintains a separate session pool for each Universal Messaging connection alias used by an audit logger. The value of this parameter must be an integer value greater than 0 and less than or equal to the value of `watt.server.audit.um.sessionPool.max`. The default value is 2.

If you specify an invalid value, Integration Server logs a warning message at start up and then uses the default value at runtime. However, Integration Server does not persist the default value over the invalid value you specified.

Note:

If you change the value of this property, you must restart Integration Server for the changes to take effect.

watt.server.audit.um.sessionPool.max

Specifies the maximum number of sessions in the Universal Messaging session pool. Integration Server maintains a separate session pool for Universal Messaging connection alias used by an audit logger. The value of this parameter must be an integer greater than 0 and greater than or equal to the value of `watt.server.audit.um.sessionPool.min`. The default value is 10.

If you specify an invalid value, Integration Server logs a warning message at start up and then uses the default value at runtime. However, Integration Server does not persist the default value over the invalid value you specified.

Note:

If you change the value of this property, you must restart Integration Server for the changes to take effect.

watt.server.audit.um.sessionPool.retryInterval

Specifies the number of seconds for Integration Server to wait between attempts to re-establish a session on the Universal Messaging server after an audit logger enters queue fail-fast mode. An audit logger enters queue fail-fast mode whenever it cannot establish a connection to the Universal Messaging server that contains the audit logging queue. Specify an integer greater than 0 (zero). The default is 30 seconds.

If you specify an invalid value, Integration Server uses the default value at runtime. However, Integration Server does not persist the default value over the invalid value you specified.

Note:

If you change the value of this property, you must restart Integration Server for the changes to take effect.

watt.server.auditDocIdField

Specifies a custom document ID value to identify documents in a standard way and to provide uniform business context in the logging display. Some documents are logged by webMethods Broker through WmLogUtil to the document database, and some are logged by various components within the Integration Server, for example, if a service fails, or if the number of retries in a trigger are exceeded. As a result, when viewing the Document Monitor, some documents are logged with a numeric document ID, and some are logged with lengthy hexadecimal strings as the document ID. The custom document ID value that you specify will be used to create the document logging ID. This value is used in place of the `BrokerEvent.getEventId()` value (the original document ID behavior). The value must be in the form of a Broker unicode string, and values in excess of 128 characters will be truncated. If this extended setting is missing, the original document ID behavior applies. If this extended setting is present but undefined (null), the `_env.uuid` value is used if present; if no `_env.uuid` value is defined, the original document ID behavior applies. For more information about document logging, see *Administering webMethods Broker*.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.auth.cache.capacity

Specifies the number of user name and password combinations Integration Server stores in the authentication cache. The default value is 250.

watt.server.auth.cache.enabled

Specifies whether the authentication cache is enabled. When set to true, the authentication cache is enabled. The default value is true.

watt.server.auth.cache.timeout

Specifies the number of milliseconds that each cache entry can remain idle before Integration Server removes it from the authentication cache. The default is 300000 (5 minutes).

Note:

Entries removed from the authentication cache are added again the next time the credentials are authenticated successfully.

Note:

Once a user has changed the password and logged in successfully with the new password, Integration Server removes the old password from the authentication cache.

watt.server.auth.checkWithSession

When Integration Server receives an HTTP request that includes both an Authorization header and a Cookie header with a session identifier, this parameter specifies whether Integration Server should check that the user identified in the Authorization header is the owner of the session identified in the Cookie header. When set to `true` (the default), Integration Server confirms that the user of the Authorization header is the owner of the identified in the Cookie header. If the headers do not match, Integration Server returns an HTTP 401 status code with the following message:

The user identified in the request does not own the requested session.

Important:

If you set `watt.server.auth.checkWithSession` to `false`, Integration Server allows HTTP requests to use mismatched credentials to access the services on Integration Server and, as a result, might put your applications at risk.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.auth.oauth.accessToken.useHeaderFields

Specifies whether Integration Server performs OAuth authentication when an inbound HTTP/S request includes an `access_token` in the header fields. Specify `true` to perform OAuth authentication. Specify `false` to skip OAuth authentication. The default is `true`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.auth.oauth.accessToken.useQueryParameters

Specifies whether Integration Server performs OAuth authentication when an inbound HTTP/S request includes an `access_token` in the query parameter. Specify `true` to perform OAuth authentication. Specify `false` to skip OAuth authentication. The default is `true`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.auth.samlResolver

Specifies the URL to the SAML artifact resolver endpoint on the My webMethods Server that will be used to validate My webMethods Server users on Integration Server. Along with configuring central user management, the value of this property allows single-sign on for My webMethods Server users. For more information, see [“Configuring the MWS Single Sign-On Resource Setting” on page 526](#).

Note: Integration Server adds this property to the `server.cnf` file when you specify a value for the **MWS SAML Resolver URL** field on the **Settings > Resources** page of Integration Server Administrator. Software AG recommends that you use the **MWS SAML Resolver URL** field to specify the My webMethods Server SAML artifact resolver endpoint instead of changing the value of the `watt.server.auth.samlResolver` property.

watt.server.auth.session.retainJaasSubject

Specifies whether Integration Server should retain authentication credentials as part of a session. When set to `true`, Integration Server retains the authentication credentials until the session expires. If set to `false` (the default), Integration Server deletes the authentication credentials from the session after it handles the initial request from the client.

watt.server.auth.skipForMediator

Specifies whether Integration Server validates requests from webMethods API Gateway to the native service. When this parameter is set to `true`, Integration Server skips authentication for API Gateway requests. When set to `false` (the default) Integration Server authenticates all API Gateway requests.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.autodeploy.enabled

Specifies whether automatic deployment of packages is enabled or disabled for Microservices Runtime. Set to `true` to enable automatic deployment of packages. Set to `false` to disable automatic deployment of packages. The default is `false`.

Note:

The automatic deployment feature is available by default for Microservices Runtime. To use the automatic deployment feature with Integration Server, your Integration Server must have additional licensing.

Important:

If you change the setting of this parameter, you must restart Microservices Runtime for the changes to take effect.

watt.server.autodeploy.interval

Specifies the interval, measured in minutes, at which Microservices Runtime executes the autodeploy system task. The autodeploy system task monitors the autodeploy folder for packages that need to be automatically deployed. Specify a value between 1 and 1440. The default is 5.

Note:

The automatic deployment feature is available by default for Microservices Runtime. To use the automatic deployment feature with Integration Server, your Integration Server must have additional licensing.

Important:

If you change the setting of this parameter, you must restart Microservices Runtime for the changes to take effect.

watt.server.autodeploy.alwaysUseHotDeployment

Specifies whether or not Microservices Runtime always uses hot deployment for automatic deployment of packages. Set to `true` to use hot deployment for auto deployment of packages. Set to `false` if you do not want to use hot deployment for automatic deployment of packages. The default is `false`.

Note:

The automatic deployment feature is available by default for Microservices Runtime. To use the automatic deployment feature with Integration Server, your Integration Server must have additional licensing.

Important:

If you change the setting of this parameter, you must restart Microservices Runtime for the changes to take effect.

watt.server.broker.producer.multiclient

Specifies the number of sessions for the default client. The default client is the Broker client that the Integration Server uses to publish documents to the Broker and to retrieve documents delivered to the default client. When you set this parameter to a value greater than 1, the Integration Server creates a new multi-session, shared state Broker client named *clientPrefix_DefaultClient_MultiPub*, to use for publishing documents to the Broker. Using a publishing client with multiple sessions can lead to increased performance because it allows multiple threads to publish documents concurrently. The default is 1 session.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.broker.replyConsumer.fetchSize

Specifies the number of reply documents that the Integration Server retrieves from the Broker at one time. Increasing the reply documents the Integration Server retrieves for each call can reduce the number of calls the Integration Server makes to the Broker. The Integration Server maintains all reply documents in memory. You can reduce the amount of memory used for reply documents by decreasing the number of documents the Integration Server retrieves at one time. The default is 5 documents.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.broker.replyConsumer.multiclient

Specifies the number of sessions for the request/reply client. The request/reply client is the Broker client that the Integration Server uses to send request documents to the Broker and to retrieve reply documents from the Broker. Increasing the number of sessions for the request/reply client can lead to improved performance because it allows multiple requests and replies to be sent and retrieved concurrently. The default is 1 session.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.broker.replyConsumer.sweeperInterval

Specifies how often (in milliseconds) the Integration Server sweeps its internal mailbox to remove expired replies to published requests. The length of the interval should balance the amount of memory consumed by expired replies with retrieving the replies for waiting requests. The Integration Server uses one background thread to age and remove expired replies and uses multiple background threads to retrieve replies for waiting requests. When the sweeper thread removes expired replies, it blocks the threads attempting to retrieve replies. When the sweeper interval is too low, the frequent execution of the sweeper thread can degrade performance.

because other background threads cannot retrieve replies as often. A sweeper interval that is too high can cause an increase in memory usage because expired replies consume memory for a longer period of time. The default is 30000 milliseconds (30 seconds).

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.brokerTransport.dur

Specifies the number of seconds of idle time that the Broker waits before sending a keep-alive message to Integration Server. If Integration Server does not respond within the amount of time specified by the `watt.server.brokerTransport.max` property, the Broker sends another keep-alive message to Integration Server. If Integration Server continues to be unresponsive, the Broker continues sending keep-alive messages until it reaches the retry limit specified by the `watt.server.brokerTransport.ret` property. If the Integration Server still has not responded to the keep-alive message, the Broker explicitly disconnects the Integration Server. The `watt.server.brokerTransport.dur` value must be an integer greater than or equal to zero but less than 2147483647. The default is 60 seconds.

For more information about using server parameters to configure the keep-alive setting with the Broker, see [“Setting Server Configuration Parameters for Keep-Alive Mode” on page 259](#).

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.brokerTransport.max

Specifies the number of seconds that the Broker waits for the Integration Server to respond to a keep-alive message. This value must be an integer between 0 and 2147483647. The default is 60 seconds.

For more information about using server parameters to configure the keep-alive setting with the Broker, see [“Setting Server Configuration Parameters for Keep-Alive Mode” on page 259](#).

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.brokerTransport.ret

Specifies the number of times the Broker re-sends keep-alive messages before disconnecting an un-responsive Integration Server. This value must be an integer between 1 and 2147483647. The default is 3 retries.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.cache.prefetchUser

Specifies the user authorized to prefetch cache service entries. The default is Administrator.

watt.server.cache.flushMins

Specifies how often (in minutes) the server sweeps the cache to remove expired cache entries and to prefetch cache service entries. The default is 10 minutes.

Note:

This configuration parameter applies to the service-results caching feature. It does not affect the caching capabilities provided by Ehcache.

watt.server.cache.gcMins

Specifies how often (in minutes) the server sweeps the cache to perform garbage collection. The default is 60 minutes.

Note:

This configuration parameter applies to the service-results caching feature. It does not affect the caching capabilities provided by Ehcache.

watt.server.cache.maxEntriesInCache

Specifies the default value for the **Maximum Entries in Cache** property for a distributed system cache used in an Integration Server cluster. The default is -1 which indicates that Integration Server does not set a default value for the **Maximum Entries in Cache** property. If you want Integration Server to set a default value for the **Maximum Entries in Cache** property, you must specify a value greater than 0.

watt.server.cachemanager.connectTimeout

Specifies the number of milliseconds a cache manager will wait while trying to connect to a server specified in the Terracotta Server Array URLs list. The default is 60000 milliseconds (i.e. 60 seconds).

Note:

Regardless of the `watt.server.cacheManager.connectTimeout` value, a cache manager will wait for a maximum of 300 seconds to connect to a server in the Terracotta Server Array. For example, if this property is set to 60000, the cache manager will wait for 60000 milliseconds to connect to any of the available server. If a connection is not established to any of the servers in 60000 milliseconds, an exception is thrown and then Integration Server takes the action specified by the `watt.server.cluster.action.errorOnStartup` parameter.

watt.server.cachemanager.logsDirectory

Specifies the location where Ehcache will write log files. The default value of this folder is *Integration Server_directory \instances\instance_name\logs\tc-client-logs*

watt.server.cachemanager.parallelThreads

Specifies the maximum number of threads that you want Integration Server to use when initializing cache managers at startup.

If the server contains a large number of cache managers, distributed cache managers, or a cache manager with a large number of caches, the registration process at start up could take a long time. To avoid this, you can increase the number of threads used to initialize cache managers by setting the `watt.server.cachemanager.parallel.threads` property.

If you want Integration Server to initialize cache managers sequentially, set the value to 1. If you want Integration Server to initialize cache managers in parallel, set the value to 2 or higher. Software AG recommends that you do not exceed 10. The default value is 5.

watt.server.centralUsers.shutdownOnError

Specifies whether the Central Users component shuts down when it encounters an error. When this property is set to `true`, the Central User component shuts down when it encounters an error. When the Central Users component shuts down, the Integration Server must be restarted.

When this property is set to `false`, the default, the Central User component does not shut down when it encounters an error.

watt.server.cgi.cache

Specifies whether the output pipeline should display all the elements in a response when using CGI content handlers to format the responses. When this property is set to `false`, the output pipeline displays all the elements in the response. When this property is set to `true`, if the response contains multiple non-string, non-null elements with the same value, the output pipeline will display only the first occurrence of the value. For each of the subsequent occurrences, the output will contain the string, `*ObjectRef(className)*`. The default is `false`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.cgi.unicode

Specifies whether the CGI content handler should use Unicode encoding. When this property is set to `false`, the CGI content handler will use the encoding specified by the `watt.server.netencoding` property to encode responses. When this property is set to `true`, the CGI content handler will use Unicode to encode responses. The default is `false`.

watt.server.checkAclsInternally

Specifies whether Integration Server performs ACL checking when a service is directly invoked by a client or trigger and when it is invoked from other services. When set to `true`, Integration Server always checks the Execute ACL for a service, regardless of the value of the Enforce Execute ACL property for the service. When set to `false`, the value of the Enforce Execute ACL property determines whether or not Integration Server performs ACL checking when a service executes. The default is `false`.

watt.server.checkPath.restorePipelineFromFile

Specifies whether Integration Server verifies that the value of the *filename* input parameter supplied to `pub.flow.restorePipelineFromFile` service is in the default pipeline directory or is in the `allowedReadPaths` parameter of the file access control configuration file (`fileAccessControl.cnf`). When this parameter is set to `true`, Integration Server verifies that the path or directory specified in *filename* is included in the default pipeline directory or in the `allowedReadPaths` parameter. If the file is not in the allowed list, the service ends with a `ServiceException`. When the `watt.server.checkPath.restorePipelineFromFile` parameter is set to `false`, Integration Server does not verify that the specified filename is in the default pipeline directory or listed in the `allowedReadPaths` parameter. The default is `true`, which is the more secure option.

watt.server.checkPath.savePipelineToFile

Specifies whether Integration Server verifies that the value of *filename* input parameter supplied for the `pub.flow.savePipelineToFile` service is in the default pipeline directory or is in the `allowedWritePaths` parameter of the file access control configuration file (`fileAccessControl.cnf`). When this parameter is set to `true`, Integration Server verifies that the path or directory specified in *filename* is in the default pipeline directory or is included in the `allowedWritePaths` parameter. If the file is not in the `allowedWritePaths` list, the service ends with a `ServiceException`. When the `watt.server.checkPath.savePipelineToFile` parameter is set to `false`, Integration Server does not verify that the specified filename is in the default pipeline directory or listed in the `allowedWritePaths` parameter. The default is `true`, which is the more secure option.

watt.server.checkWhitelist

Specifies whether Integration Server uses a whitelist to filter the list of classes than can be used for deserialization. When this parameter is set to true, whitelist filtering is enabled. Integration Server deserializes a Java object only if the class appears in the whitelist. If Integration Server encounters a Java object whose class does not exist in the whitelist, Integration Server throws a `ClassNotFoundException`. By default, `watt.sever.checkWhitelist` is set to true, indicating whitelist class filtering is enabled.

For more information about whitelist filtering, see [“Whitelist Filtering in Integration Server” on page 925](#).

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.circuitBreaker.threadPoolMax

Specifies the maximum number of threads that the server maintains in the circuit breaker thread pool which is used to execute services with a configured circuit breaker. If this maximum number is reached, the server waits until services complete and return threads to the circuit breaker thread pool before running more services with a configured circuit breaker. You must specify a value greater than 0 and greater than or equal to the value of `watt.server.circuitBreaker.threadPoolMin`. The default is 75.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

The circuit breaker feature is available by default for a service that resides in a Microservices Runtime. To use the circuit breaker feature with Integration Server, your Integration Server must have additional licensing.

watt.server.circuitBreaker.threadPoolMin

Specifies the minimum number of threads that the server maintains in the circuit breaker thread pool which is used to executes services with a configured circuit breaker. When the server starts, the circuit breaker thread pool initially contains this minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed, which is specified by the `watt.server.circuitBreaker.threadPoolMax`. You must specify a value greater than or equal to 0 (zero) and less than or equal to the value of `watt.server.circuitBreaker.threadPoolMax`. The default is 10.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

The circuit breaker feature is available by default for a service that resides in a Microservices Runtime. To use the circuit breaker feature with Integration Server, your Integration Server must have additional licensing.

watt.server.classloader.pkgpriority

Specifies the order in which Integration Server scans packages when loading a class file for which no package information is available. When no order is specified, Integration Server attempts to load the class by scanning all packages in the Packages directory, one by one. This could delay class loading. For example, when a trigger contains a reference to a Java class, the trigger does not have any package information. Integration Server scans all the packages for the specific class file in random order. Specifying the packages that Integration Server should scan first may reduce the time needed for class loading.

Use the `watt.server.classloader.pkgpriority` property to specify a comma delimited list of the packages whose class loader should load first. The syntax for this property is:

```
watt.server.classloader.pkgpriority=packageName, packageName
```

For more information about class loading, refer to [“Class Loading in Integration Server”](#) on page 911.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.clientTimeout

Specifies the amount of time (in minutes) after which an idle user session times out. The default is 10.

watt.server.clientTimeoutRedirect

Specifies how Integration Server Administrator behaves when the browser session times out. When this parameter is set to `true`, Integration Server Administrator redirects you to a blank page with an option to resume, which prompts you to log in. On successful login, the home page is displayed. When set to `false`, which is also the default value, Integration Server Administrator prompts you to log in, while retaining the current page contents in the background. On successful login, you can resume from the same page.

Note:

This parameter was introduced for PIE-64200 in IS_10.5_Core_Fix6.

watt.server.cluster.action.errorOnStartup

Specifies how Integration Server responds when an error at start up prevents Integration Server from joining the cluster. Select one of the following:

Value	Description
shutdown	<p>If Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server shuts down immediately. To make any changes to the clustering configuration, you must start Integration Server in safe mode, then edit the configuration, and then restart Integration Server.</p> <p>shutdown corresponds to the Action on Startup Error parameter option Shut Down Integration Server on the Settings > Clustering > Edit screen.</p>

Value	Description
standalone	<p>If Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server starts as a stand-alone, unclustered Integration Server. Integration Server continues to receive requests on inbound ports and serve requests. Integration Server does not use any distributed caches and instead uses local caches. You can use Integration Server to make changes needed to resolve the startup errors. When you restart Integration Server, Integration Server will rejoin the cluster if Integration Server does not encounter any start up errors that prevent the connection to the Terracotta Server Array.</p> <p>This is the default value.</p> <p>standalone corresponds to the Action on Startup Error parameter option Start as Stand-Alone Integration Server on the Settings > Clustering > Edit screen.</p>
quiesce	<p>If Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server starts as a stand-alone, unclustered Integration Server in quiesce mode. When Integration Server is in quiesce mode, only the diagnostic port and quiesce port are enabled. Integration Server will not receive requests on inbound ports. However, you can use Integration Server and Integration Server Administrator to make changes needed to correct the startup errors. After correcting the errors, exit quiesce mode. When exiting quiesce mode, Integration Server starts all the cache managers and joins the cluster.</p> <p>If Integration Server encounters any errors that prevent it from joining the cluster when exiting quiesce mode, Integration Server does not re-enter quiesce mode. Instead, Integration Server runs as an unclustered, stand alone server. Integration Server captures any errors that occurred while attempting to join the cluster in a report displayed in Integration Server Administrator when the server exits quiesce mode. Use the report and error logs to help correct the underlying configuration errors. Then, to rejoin the cluster, either restart Integration Server or enter and then exit quiesce mode.</p> <p>To use the quiesce option, a quiesce port must be configured for Integration Server. If a quiesce port is not configured and the quiesce option is selected, when Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server shuts down instead of entering quiesce mode.</p> <p>quiesce corresponds to the Action on Startup Error parameter option Enter Quiesce Mode on Stand-Alone Integration Server on the Settings > Clustering > Edit screen.</p>

Value	Description
-------	-------------

For more information about quiesce mode, see [“Quiescing the Server for Maintenance” on page 929](#).

The `watt.server.cluster.action.errorOnStartup` parameter displays the value specified in the **Action On Startup Error** parameter on the **Settings > Clustering** and **Settings > Clustering > Edit** screens of Integration Server Administrator. Software AG recommends that you specify the action to take when Integration Server cannot connect to the cluster on startup using the **Action On Startup Error** parameter on the **Settings > Clustering > Edit** screen instead of by changing the value of the `watt.server.cluster.action.errorOnStartup` parameter.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.cluster.aliasList

Specifies a comma-delimited list of aliases for remote Integration Servers in a cluster. The Integration Server uses this list when executing the remote invokes that update the other cluster nodes with trigger management changes. When this property is configured, the **Settings > Messaging > webMethods Messaging Trigger Management > Cluster View** screen will be visible and the **Apply Change Across Cluster** check box will be available when performing trigger management tasks.

This parameter is applicable only when you are using webMethods clustering. For more information, see the *webMethods Integration Server Clustering Guide*.

watt.server.cluster.aware

Specifies whether you want the server to participate in a cluster. A value of `true` indicates that clustering is enabled for the server. The default is `false`.

The value of the `watt.server.cluster.aware` parameter is tied to the value of the **Clustering Status** field on the **Settings > Clustering** screen in Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to enable or disable clustering for an Integration Server instead of using the `watt.server.cluster.aware` property.

This parameter is applicable only when you are using this Integration Server in a cluster.

Important:

You must restart Integration Server for changes to take effect.

Note:

If `watt.server.cluster.aware` is set to `true`, you must also specify values for the `watt.server.cluster.tsaURLs` and `watt.server.cluster.name` parameters.

watt.server.cluster.name

Specifies the name of the cluster to which Integration Server belongs. If Integration Server does not belong to a cluster, this parameter is empty.

Keep the following in mind when specifying the cluster name:

- The cluster name cannot include any periods “.”. Integration Server converts any periods in the name to underscores when you save the cluster configuration.

- The cluster name cannot exceed 32 characters. If it does, Integration Server uses the first 20 characters of the supplied name and then a hash for the remaining characters.

The `watt.server.cluster.name` parameter displays the value specified in the **Cluster Name** parameter on the **Settings > Clustering** and **Settings > Clustering > Edit** screens of Integration Server Administrator. Software AG recommends that you specify the cluster name using the **Cluster Name** parameter on the **Settings > Clustering > Edit** screen instead of by changing the value of the `watt.server.cluster.name` parameter.

Important:

You must restart Integration Server for changes to take effect.

Note:

If `watt.server.cluster.aware` is set to `true`, you must also specify values for the `watt.server.cluster.tsaURLs` and `watt.server.cluster.name` parameters.

watt.server.cluster.SessTimeout

Specifies number of minutes that the server allows inactive session objects to remain in the cluster store before removing them. The default is 60.

This parameter is applicable only when you are using webMethods Integration Server Clustering. For more information, refer to the *webMethods Integration Server Clustering Guide*.

watt.server.cluster.tsaURLs

A comma-separated list of the Terracotta Server Array URLs that the cluster uses for distributed system caches. This parameter is applicable only when Integration Server belongs to a cluster.

The `watt.server.cluster.tsaURLs` parameter is tied to **Terracotta Server Array URLs** parameter on the **Settings > Clustering** and **Settings > Clustering > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the Terracotta Server Array URLs using the **Terracotta Server Array URLs** parameter on the **Settings > Clustering > Edit** screen instead of by changing the value of the `watt.server.cluster.tsaURLs` parameter.

Important:

You must restart Integration Server for changes to take effect.

Note:

If `watt.server.cluster.aware` is set to `true`, you must also specify values for the `watt.server.cluster.tsaURLs` and `watt.server.cluster.name` parameters.

watt.server.coder.bincoder.trycontextloaderfirst

Specifies whether Integration Server uses the context loader before using the class loader for the currently executing thread when Integration Server encodes or decodes a pipeline. If a referenced class belongs to a particular package, it may be faster to use the context loader first. The default is `false`.

watt.server.coder.responseAsXML

Specifies how Integration Server receives the XML response for any REST API request. When this property is set to `true`, Integration Server receives the XML response for any REST API in proper XML format. When this property is set to `false`, Integration Server receives the XML response for any REST API in converted XML format containing `IData` objects. The default is `true`.

watt.server.commonmessaging.connection.retryPeriod

Specifies the length of time, in seconds, that Integration Server waits between connection attempts when a connection to the MQTT server fails. The default is 20 seconds.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.commonmessaging.trigger.monitoringInterval

Specifies the interval, measured in seconds, at which Integration Server executes resource monitoring services for MQTT triggers. A resource monitoring service is a service that you create to check the availability of resources used by an MQTT trigger service. After Integration Server stops an MQTT trigger because all retry attempts have failed, it schedules a system task to execute the resource monitoring service assigned to the MQTT trigger. The default is 60 seconds.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.commonmessaging.trigger.restartTaskRetryCount

Specifies the maximum number of retry attempts the trigger restart task makes to start MQTT triggers automatically. Integration Server makes the initial start attempt for an MQTT trigger when starting the MQTT connection alias used by the trigger. If any MQTT triggers fail to start when the alias starts, Integration Server uses a trigger restart task to retry starting the failed MQTT triggers. The restart task, which uses a separate thread from the thread used to start the alias, runs at an interval specified by the `watt.server.commonmessaging.trigger.restartTaskRetryInterval` parameter. Integration Server also uses a restart task when triggers fail at run time. For example, if the MQTT connection alias becomes unavailable, the MQTT trigger eventually fails.

To use a restart task, you must specify a positive integer greater than or equal to 1. The default is 6 retry attempts. Set `watt.server.commonmessaging.trigger.restartTaskRetryInterval` to 0 if you do not want Integration Server to attempt to restart MQTT triggers automatically.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.commonmessaging.trigger.restartTaskRetryInterval

Specifies the number of seconds that the trigger restart task waits between attempts to restart MQTT triggers. You must specify a positive integer. The default is 30 seconds.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.commonmessaging.trigger.reuseSession

Indicates whether instances of an MQTT trigger use the same session on Integration Server. When this property is set to true, the MQTT trigger uses a shared session. Each of the trigger services executed by the MQTT trigger will use the same session ID. When this property is set

to `false`, Integration Server uses a new session for each instance of a MQTT trigger. Reusing sessions for an MQTT trigger might improve performance. The default is `false`.

watt.server.commonmessaging.trigger.stopRequestTimeout

Specifies the maximum amount of time, measured in seconds, that Integration Server waits after an MQTT trigger is disabled before forcing the MQTT trigger to stop processing messages. The minimum value is 0. The default is 3 seconds.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.compile

Specifies the compiler command that Integration Server uses to compile Java services that are developed using Designer, for example, `javac -classpath {0} -d {1} {2}`. This compiler command is also used from the `jcode` utility. If this property is omitted or empty, the server uses the JVM internal Java compile tool to compile Java services.

Note:

If you use the `watt.server.compile` property to have Integration Server use a different compiler, and that compiler returns errors and warnings to standard out rather than standard error, set `watt.server.compile.readFromStdErr` to `false` so that the compiler errors and warnings will be read and displayed to the Java developer.

watt.server.compile.exitOnError

Specifies whether the `jcode` utility should stop processing after it fails. If set to `true`, the `jcode` utility stops processing at the package on which it fails. If set to `false` (the default), the `jcode` utility will continue processing despite the failure.

watt.server.compile.readFromStdErr

Specifies from where Integration Server reads compiler errors and warnings. The Software AG webMethods platform includes `javac`, the Java tool used to compile Java services in Designer and with the `jcode` utility. The `javac` compiler returns information about any errors or warnings it encountered during compilation to standard error. If you use the `watt.server.compile` property to have Integration Server use a different compiler, and that compiler returns errors and warnings to standard out rather than standard error, set `watt.server.compile.readFromStdErr` to `false` so that the compiler errors and warnings will be read and displayed to the Java developer. Set `watt.server.compile.readFromStdErr` to `true` for Integration Server to read compiler errors and warnings from standard error. The default is `true`.

watt.server.compile.unicode

Specifies the compiler command that Integration Server uses to compile Java services that are stored in Unicode encoding, for example, `javac -encoding Unicode -classpath {0} -d {1} {2}`. Note that the setting “`javac -encoding Unicode -classpath {0} -d {1} {2}`” works with the Oracle JDK compiler. This compiler command is also used from the `jcode` utility. If this property is omitted or empty, the server uses the JVM internal Java compile tool to compile Java services.

watt.server.content.type.default

Specifies what content type to use for HTTP requests that do not contain a Content-Type header. The default value is `application/octet-stream`.

Note:

If you do not specify a value for `watt.server.content.type.default` and the Content-Type header is empty, Integration Server assigns the default value of `application/octet-stream`. In this case, you must create a content handler for `application/octet-stream`.

To specify that Integration Server should use a different Content-Type, change the value of `watt.server.content.type.default` to a different registered content type, such as `text/xml` or `text/html`. If you set the value of `watt.server.content.type.default` to an unregistered content type, Integration Server treats requests with a Content-Type header as if they are `text/html`. Changes to this property take effect immediately.

watt.server.content.type.mappings

Specifies a one-to-one mapping of content-types, including mapping wild-carded content types to a specific content type.

One use of this parameter is for mapping wildcards found in the Accept header field of an HTTP client request to specific content types. The Accept header field specifies which content type or types the client will accept in a response. Integration Server selects an outbound content handler based on the allowed content type.

Another use of this parameter is for specifying the content handler to use for particular content type in an inbound request.

The syntax for this parameter is:

```
watt.server.content.type.mappings=<wildcard> <content-type>,<wildcard> <content-type>,<wildcard>
<content-type> ...
```

Where:

wildcard is a wild-carded content type, such as `text/*`, or the content type without a wildcard character

content-type is the specific content type to use for the wild-carded content type or the specified content type without a wildcard character.

Use a space to separate the wildcarded content type from the specific content type. Use a comma to separate `<wildcard> <content-type>` pairs from other pairs. Consider the following examples:

To associate `text/*` with `text/xml`, specify:

```
watt.server.content.type.mappings=text/* text/xml
```

To associate `text/*` with `text/xml`, and `multipart/*` with `multipart/related`, specify:

```
watt.server.content.type.mappings=text/* text/xml,multipart/* multipart/related
```

Integration Server parses the Accept header and attempts to select a content type or types as defined in the W3C Hypertext Transfer Protocol specification.

The default setting is `text/* text/xml`. If this parameter is not specified, and an Accept header field specifies a wildcard, Integration Server selects a content handler based on the major content type specified, as shown below.

This Accept Header field	Results in this content handler
<code>text/*</code>	XML
<code>application/*</code>	CGI

This Accept Header field

multipart/*

Results in this content handler

Multipart

The `watt.server.http.useAcceptHeader` property must be set to true for the `watt.server.content.type.mappings` parameter to work.

To use `watt.server.content.type.mappings` to map the `application/xml` content type to the `text/xml` content type so that Integration Server uses the `text/xml` content handler for an inbound request instead of the `application/xml` content handler, specify the following:

```
watt.server.content.type.mappings=application/xml text/xml
```

For more information about content handlers, refer to [“Content Handlers in Integration Server” on page 901](#).

watt.server.control.controlledDeliverToTriggers.delayIncrementInterval

Specifies the interval used in conjunction with

`watt.server.control.controlledDeliverToTriggers.delays` to determine the delay that Integration Server applies to services publishing documents locally when the trigger document store reaches 90% of its capacity. The default is 2000.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.control.controlledDeliverToTriggers.delays

Specifies a comma-separated list of values specifying the number of milliseconds that Integration Server delays services publishing local documents when the trigger document store is greater than or equal to 90% of its capacity. Integration Server changes the delay based on how long the trigger document store has continuously been at 90% or higher capacity. The default value is 500,2000,3000,3500,4000,4500,5000.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.control.controlledDeliverToTriggers.pctMaxThreshold

Specifies the trigger queue threshold at which the Integration Server slows down the delivery rate of locally published documents. This threshold is expressed as a percentage of the trigger queue capacity. For example, if you specify 80, the Integration Server decreases the rate at which it delivers locally published documents to a trigger queue when that trigger queue reaches 80% capacity. Integration Server resumes delivering documents at the normal rate when the trigger queue capacity drops below the specified threshold. The default is 90.

watt.server.control.freeMemoryThreshold

Specifies the threshold (as a percentage) at which Integration Server starts to warn of insufficient free memory. When the percentage of available free memory falls below the value of this property, Integration Server generates a warning in the server log. Valid values are 0 to 100. The default is 5.

watt.server.control.maxPersist

Specifies the capacity of the outbound document store. Integration Server places published documents in the outbound document store when the configured Broker is unavailable. When the number of documents in the outbound document store equals the capacity, the Integration Server blocks any threads executing services that publish documents. The Integration Server resumes execution of blocked threads after the Broker becomes available. Set this parameter to a whole number greater than 0. The default is 500,000 documents.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.control.maxPublishOnSuccess

Specifies the maximum number of documents that the server can publish on success at one time. For example, suppose that you set the maximum to 100 documents. ServiceA publishes 10 documents on success. ServiceB publishes 90 documents on success. ServiceC publishes 5 documents on success. ServiceA and ServiceB can publish documents concurrently. However, if ServiceC begins to publish documents before ServiceA or ServiceB completes, the Integration Server throws an exception for ServiceC because the documents published by ServiceC exceed the maximum number of documents that can be published on success at one time. If ServiceD publishes 125 documents on success and the maximum is 100, ServiceD will receive an exception every time it executes. The default is 50,000 documents.

watt.server.control.memorySensorInterval

The time interval (in milliseconds) at which the Integration Server will check the available free memory. The default is 300000 (5 minutes).

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.control.serverThreadThreshold

Specifies the threshold at which Integration Server starts to warn of insufficient available threads. When the percentage of available server threads goes below the value of this property, Integration Server generates a journal log message indicating the current available thread percentage and stating "Available Thread Warning Threshold Exceeded." When you receive this message in the journal log, you can adjust the thread usage to make server threads available.

The default is 15%.

Note:

It is recommended that you use the **Available Threads Warning Threshold** field in the Edit Resource Settings page of the Integration Server Administrator to set the threshold value. For more information about setting an available threads warning threshold, see ["Managing the Server Thread Pool" on page 109](#).

watt.server.control.threadSensorInterval

Specifies the interval, measured in milliseconds, at which Integration Server will monitor the thread usage. When the percentage of available server threads goes below the threshold value specified in the watt.server.control.serverThreadThreshold property, Integration Server will generate a journal log message to warn of insufficient available threads. The default is 2000 milliseconds.

watt.server.control.triggerInputControl.delayIncrementInterval

Specifies the interval used in conjunction with `watt.server.control.triggerInputControl.delays` to determine the frequency with which Integration Server polls a trigger client queue on the Broker if the queue was empty when it was last polled. The default is 10000.

Note:

If an exception occurs when parsing the supplied values for `watt.server.control.triggerInputControl.delays` or `watt.server.control.triggerInputControl.delayIncrementInterval`, Integration Server resets both configuration parameters to the default values.

Note:

This parameter is deprecated because `webMethods Broker` is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

For more information about controlling the polling frequency for trigger client queues on the Broker, see [“Delaying Polling Requests for webMethods Messaging Triggers”](#) on page 853.

watt.server.control.triggerInputControl.delays

Specifies a comma-separated list of values specifying the number of milliseconds that Integration Server delays the polling of a trigger client queue on the Broker. Integration Server changes the delay based on how long the trigger client queues has been empty. The default value is 500,1000,1500,5000.

Note:

If an exception occurs when parsing the supplied values for `watt.server.control.triggerInputControl.delays` or `watt.server.control.triggerInputControl.delayIncrementInterval`, Integration Server resets both configuration parameters to the default values.

Note:

This parameter is deprecated because `webMethods Broker` is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

For more information about controlling the polling frequency for trigger client queues on the Broker, see [“Delaying Polling Requests for webMethods Messaging Triggers”](#) on page 853.

watt.server.cors.allowedOrigins

Specifies the URIs from which Integration Server is to allow cross-origin requests to access resources.

As recommended by the CORS (Cross-Origin Resource Sharing) standard, Integration Server uses the values of this parameter to validate the request. Integration Server checks the URI specified in the Origin header of the CORS request. If it does not match a URI specified by this parameter, Integration Server refuses the request and returns an HTTP 403 status code with the following message:

Specified Origin is not allowed

If the URI matches a value specified by this parameter, Integration Server sends a response with the CORS Access-Control-Allow-Origin response header back to the requesting user agent.

You can set this parameter to a comma- or space-separated list of URIs or the absolute path to a file. You cannot specify a combination of these.

To define this parameter using a comma- or space-separated list of URIs, enter *protocol://hostname* or *protocol://hostname:portnumber*. You can specify the IP address or the name of the machine for *hostname*. The values are case-sensitive. Use a space or comma delimiter to specify multiple values.

To define this parameter using a text file set the value to:

```
file:pathToFile\filename
```

For example: `watt.server.cors.allowedOrigins=file:c:\cors\allowedOriginsList.txt`

The file must use the same syntax as is required for the `watt.sever.cors.allowedOrigins` with the difference that each line must be an allowed origin server URI or a regular expression.

You can invoke the following URL to execute the internal service that causes the referenced text file to be loaded into Integration Server:

```
http://host:port/invoke/wm.server.admin:refreshAllowedOrigins
```

Note:

The IP address and host name cannot be used interchangeably. If you specify a host name, and a cross-origin request is received that uses the corresponding IP address or vice versa, Integration Server will reject the request.

You can use an asterisk (*) to indicate that any URI or origin is allowed.

Note:

You cannot use the asterisk to indicate URIs if `watt.server.cors.supportsCredentials` is set to `true`.

You can also use regular expressions in the comma-separated list of allowed origin servers which can make the list of origin servers easier to maintain. Integration Server treats any value in the comma-separated list that begins with "r:" as a regular expression. Integration Server treats any value that does not begin with "r:" as a simple string. The server configuration parameter uses the Java regular expression syntax, as documented at <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>. A regular expression value must match the entire value of the Origin header in the HTTP request for it to be considered a match.

Example:

```
watt.server.cors.allowedOrigins=http://test1.domain.com,r:https?://.*.test2.domain.com:[0-9]+,r:.\.[a-zA-Z]*-int.domain.com
```

Integration Server treats the first value, "http://test1.domain.com", as a simple string. If an Origin header contains this value, it will be allowed.

The second value, "r:https?://.*.test2.domain.com:[0-9]+", contains a regular expression. The "r:" is not part of the regular expression. The actual regular expression used to match supplied Origin headers is "https?://.*.test2.domain.com:[0-9]+".

The third value, "r:.\.[a-zA-Z]*-int.domain.com", contains a regular expression. The "r:" is not part of the regular expression. The actual regular expression used to match supplied Origin headers is ".\.[a-zA-Z]*-int.domain.com".

"Origin: http://test1.domain.com" will be allowed because it is equal to the first value.

"Origin: http://my.test2.domain.com:8080" will be allowed because it matches the second value.

"Origin: https://my.test2.domain.com:8088" will be allowed because it matches the second value.

"Origin: http://my.test2.domain.com" will not be allowed. If it had a port number, it would match the second value.

"Origin: nbps://example.prod-int.domain.com" will be allowed because it matches the third value.

"Origin: example.qa.staging-int.domain.com" will be allowed because it matches the third value.

"Origin: example.dev1-int.domain.com" will not be allowed. If the second token of the host name did not include any digits, it would have matched the third value.

Regular expressions that match any host name, IP address and port (e.g. "r:." and "r:.*") will have the same effect as "*".

Note:

When CORS is enabled, Integration Server evaluates the list of regular expressions in `watt.server.cors.allowedOrigins` sequentially for every request. Integration Server performs a regular expression match operation on each regular expression until a match is found or all regular expressions in the list have been evaluated. Software AG recommends that you put the more frequently matched regular expressions at the beginning of the comma-separated list.

Note:

You must specify a value if the `watt.server.cors.enabled` parameter is set to `true`.

watt.server.cors.enabled

Specifies whether Integration Server is to support CORS. If this parameter is set to `true`, Integration Server allows cross-origin requests. The default value is `false`.

Note:

In a webMethods Enterprise Gateway configuration, enable support for CORS only on the Enterprise Gateway Server.

For more information about using CORS with Integration Server, see [“Using CORS with Integration Server” on page 135](#). For instructions on how to configure Integration Server to allow cross-origin requests, see [“Configuring Integration Server to Accept CORS Requests” on page 135](#).

watt.server.cors.exposedHeaders

Specifies the values Integration Server can include with the CORS Access-Control-Expose-Headers header in response to a CORS request. The CORS Access-Control-Expose-Headers header defines the headers that can be exposed to the user agent. Examine your client-side code to determine which response headers, if any, are retrieved by the client and need to be exposed.

The value for this parameter is not case-sensitive. Use a space or comma delimiter to specify multiple values. For example:

```
watt.server.cors.exposedHeaders=<value1,value2,value3>
```

As determined by the CORS (Cross-Origin Resource Sharing) standard, the following are considered simple response headers and are always exposed: Cache-Control, Content-Language, Content-Type, Expires, Last-Modified, Pragma.

watt.server.cors.host

Specifies the host and port on which clients send cross-origin requests to Integration Server. Integration Server uses this value to validate the Host header in the cross-origin request. For improved security, Software AG recommends that you define this parameter if support for CORS is enabled (that is, `watt.server.cors.enabled` is set to `true`).

To define this parameter, enter `hostname:port` or `hostname`. You can specify the IP address or the name of the machine for `hostname`. The value for this parameter is case-sensitive. If a value is not provided for this parameter, Integration Server does not validate the Host header in the request. If a value is provided but does not match the Host header in the request, Integration Server returns a 403 response with the message.

Note:

The IP address and host name cannot be used interchangeably. If you specify a host name, and a cross-origin request is received that uses the corresponding IP address or vice versa, Integration Server will reject the request.

watt.server.cors.maxAge

Specifies the amount of time in seconds a user agent is allowed to cache the results of a preflight request. Integration Server uses this value in the CORS Access-Control-Max-Age response header.

The default is `-1`, which indicates the results do not remain on the client. The value must be an integer.

watt.server.cors.supportsCredentials

Specifies whether Integration Server is to set the CORS Access-Control-Allow-Credentials header in response to all CORS requests.

When set to `true`, Integration Server sets the CORS Access-Control-Allow-Credentials header to `true` in the response. When setting this parameter to `true`, keep the following points in mind:

- If the user credentials are valid and the user is authorized to access the resources, Integration Server will execute a CORS simple request with credentials. The user agent will display the response to the user.
- When Integration Server responds to a CORS preflight request, the response includes the Access-Control-Allow-Credentials header set to `true` so the user agent will allow the request with credentials.

Note:

User credentials can be cookies, HTTP authentication, or client-side SSL certificates.

When set to `false` (the default), Integration Server does not set the CORS Access-Control-Allow-Credentials header in the response. When setting this parameter to `false`, keep the following points in mind:

- If the user credentials are valid and the user is authorized to access the resources, Integration Server will execute a CORS simple request with credentials. The user agent will not display the response to the user.
- When Integration Server responds to a CORS preflight request, the response does not include the Access-Control-Allow-Credentials header, so the user agent will not allow the request with credentials.

Note:

For security reasons, Software AG recommends using the default setting (`false`) for this parameter. When set to `false`, the user agent ignores the cookies sent in the response. When this parameter is set to `true`, Software AG highly recommends using CSRF protection to safe guard your resources. For a more secure approach, consider using another standard, such as OAuth, for accessing authorized resources instead of relying on credentialed requests.

watt.server.cors.supportedHeaders

Specifies the request headers Integration Server will allow in cross-origin requests. Integration Server uses the value to validate the CORS Access-Control-Request-Header request header. If the header is not an exact match of a value specified by `watt.server.cors.supportedHeaders`, Integration Server refuses the request and returns an HTTP 403 status code with the following message:

```
Header not supported
```

If the header matches the value specified by this parameter, Integration Server responds with the CORS header `Access-Control-Allow-Headers` populated with the value of this parameter.

Examine your server-side code to determine which request headers, if any, are read by your server application and need to be explicitly allowed.

The value for this parameter is case-sensitive. Use a space or comma to separate multiple request headers. For example,

```
watt.server.cors.supportedHeaders=<header1,header2,header3>
```

As determined by the CORS (Cross-Origin Resource Sharing) standard, the following are considered simple request headers and are always exposed: `Accept`, `Accept-Language`, `Content-Type` (when the value is `application/x-www-form-urlencoded`, `multipart/form-data`, or `text/plain`).

watt.server.cors.supportedMethods

Specifies the HTTP methods Integration Server will allow in cross-origin requests. Integration Server uses this value to validate the value of the CORS Access-Control-Request-Method request header. If the header is not an exact match of a value specified by `watt.server.cors.supportedMethods`, Integration Server refuses the request and returns an HTTP 403 status code with the following message:

```
Method not supported
```

If the header matches a value specified by this parameter, Integration Server responds with the CORS header `Access-Control-Allow-Headers` populated with the value of this parameter.

Use a space or comma to separate multiple HTTP methods. Possible values are `OPTIONS`, `HEAD`, `GET`, `POST`, `PUT`, `PATCH`, and `DELETE`. Integration Server accepts any of these values by default.

watt.server.createPackage.ignorePattern

Specifies a semicolon delimited list of the file types you do not want Integration Server to include when publishing packages. You can use this parameter to specify the file types to exclude when publishing packages through replication and deploying packages. For example, to prevent Integration Server from publishing `.svn` files, you would specify the parameter as `watt.server.createPackage.ignorePattern=.svn`.

You can specify several file types using a semi-colon (;) as a delimiter, as follows:

```
watt.server.createPackage.ignorePattern=.svn;.java;.xml
```

watt.server.cronMaxThreads

The maximum number of threads maintained for the cronjob-based threadpool, which Integration Server uses for scheduled system tasks. If this maximum number is reached, Integration Server waits until processes complete and return threads to the pool before running more processes. The default is 5.

watt.server.cronMinThreads

The minimum number of threads maintained for the cronjob-based threadpool, which Integration Server uses for scheduled system tasks. When Integration Server starts, the thread pool initially contains this minimum number of threads. The default is 2.

watt.server.dateStampFmt

Specifies the date format to use in the log files. For the server log, this parameter specifies how the date format is displayed in the server log file as well as in Integration Server Administrator. For all other logs, such as the service, error, session, guaranteed delivery, and security logs, this parameter specifies how the date format is displayed only in Integration Server Administrator.

Note:

In order to use this parameter to specify the date format for the service, error, session, guaranteed delivery, and security log files, the `watt.server.audit.displayLogs.convertTime` must be set to `true`. For usage information, see `watt.server.audit.displayLogs.convertTime`.

To specify the date format to use, you can use any format that is supported by the Java class `java.text.SimpleDateFormat`. For example, to display the date with the format 08-12-02 14:44:33:1235, specify `dd-MM-yy HH:mm:ss:SSSS`.

watt.server.date.SuppressPatternError

Specifies how the server should respond if no input is passed to the `pub.date:dateTimeFormat` service. When set to `true`, the server simply returns a null value for the *value* parameter. The default is for the server to throw an exception.

watt.server.db.blocktimeout

Specifies the maximum time in milliseconds the server is to block a request when waiting for a connection to a database. (The database must be defined by an alias in the `WmDB` package.) The default is to wait indefinitely. Specifying `-1` also means to wait indefinitely. This property is global to all pools.

watt.server.db.connectionCache

Specifies how the server manages connections to a database.

Specifying `server` tells the server to maintain a *pool* of connections for each database that is defined to the server through an alias. If a request cannot be satisfied because the pool has reached its maximum number of connections, the server blocks the request and tries again later.

Specifying `session` tells the server to create a database connection per session. That is, when the server receives a service request that requires a database connection, it will create a new connection if one doesn't already exist for that session; otherwise the server will use the connection that was previously created for that session. If the attempt to create a connection for the session fails, for example because the database has no available slots for connections, the request fails. The default is `session`.

Although enabling database connection pooling creates a pool for each database defined to your server, you can control the characteristics of each pool *individually* by using the **Edit Alias**

Information screen of the Integration Server Administrator. See the *WmDB User's Guide* for more information about configuring the server to connect to a database.

watt.server.db.maintainminimum

Specifies how the server handles purging inactive connections that have timed out. With the parameter set to `false`, the server purges all of these connections. With the parameter set to `true`, the server purges these connections, but stops when a minimum number of connections in the pool has been reached. You specify the minimum when you define the alias. This property is global to all pools.

watt.server.db.share.ISInternal

Indicates whether the Integration Server ISInternal Functional Alias on the Integration Server Administrator JDBC Pools page refers to the same database that other Integration Servers use, even though the Integration Servers are not clustered. Set this property to `true` to indicate that the Integration Server shares the same ISInternal database. Setting the property to `true` is necessary if you want to use guaranteed delivery on multiple Integration Servers that are not clustered but share the ISInternal database. You must set this property to `true` for all Integration Servers that share the ISInternal database. Set this property to `false` if the Integration Server does not share the ISInternal database with other Integration Servers. The default is `false`.

Important:

You must restart Integration Server for changes to take effect.

watt.server.db.testSQL

Specifies a SQL statement that can be used to test a database connection in a JDBC connection pool. If the `watt.server.db.testSQL` parameter specifies a SQL statement, when a caller requests a JDBC connection from a JDBC connection pool, Integration Server executes the specified SQL statement against the database used with the JDBC connection pool. If the SQL statement executes successfully, Integration Server considers the database connection to be valid and returns the connection to the caller. If the SQL statement does not execute successfully, Integration Server considers the database connection to be invalid and removes it from the JDBC connection pool. Integration Server then creates a new JDBC connection, places it in the JDBC connection pool, and returns the new connection to the caller. If no value is specified for `watt.server.db.testSQL`, Integration Server does not test database connections in the connection pool when a caller requests a JDBC connection. There is no default value for the parameter.

Note:

Testing a database connection each time a JDBC connection is requested may impact performance as each test executes a SQL statement against the database.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.debugFIPS

Specifies whether Integration Server checks the FIPS encryption provider every time a user is authenticated. When this parameter is set to `true`, every time a user is authenticated, Integration Server checks the FIPS encryption provider by using the provider to create a `java.security.MessageDigest` instance. The provider check writes the name of the provider and the generated `MessageDigest` to standard error. When set to `false`, Integration Server does not check the encryption provider every time a user is authenticated. The default is `false`.

Important:

You must restart Integration Server for changes to take effect.

watt.server.defaultContentHandler

Specifies to which content handlers Integration Server should register the text/html content type. If set to `true` (the default), then Integration Server registers the text/html content type with the `ContentHandler_Default` content handler. If set to `false`, then Integration Server registers the text/html content type with the `ContentHandler_CGI` content handler.

Important:

You must restart Integration Server for changes to take effect.

watt.server.defaultCountry

Specifies the default country, a component of a Java Locale, that Integration Server uses when selecting strings from `ResourceBundles` to return to a client. Any ISO country code is a valid value. The default value for this property is `US`. The default locale is `en-US`.

Important:

You must restart Integration Server for changes to take effect.

watt.server.defaultLanguage

The default language, a component of a Java Locale, that Integration Server uses when selecting strings from `ResourceBundles` to return to a client. Any ISO language code is a valid value. The default value for this property is `en`. The default locale is `en-US`.

Important:

You must restart Integration Server for changes to take effect.

watt.server.defaultPackage

Specifies the name of the default package. If this value is not set, Integration Server uses `WmRoot` as the default package.

Note:

If you start Integration Server in safe mode, `WmRoot` becomes the default value regardless of the value of `watt.server.defaultPackage`.

watt.server.deprecatedExceptionLogging

Specifies how the Integration Server logs service exceptions to the Integration Server error log. Set this parameter to `true` for basic exception logging or `false` (the default setting) for more detailed exception logging to help pinpoint the source of the exception.

When set to	Integration Server
<code>true</code>	<ul style="list-style-type: none"> ■ Logs the exception for each service in a nested stack as the exception moves up the stack. If a parent service catches the exception and does not rethrow it, the Integration Server still logs the exception for each child service it passed on its way up the stack. ■ Displays the stack trace for each service.

When set to	Integration Server
false	<ul style="list-style-type: none"> <li data-bbox="365 262 1385 367">■ Logs the exception only once, at the top-level service, even if the exception occurred at a nested service. If any service in the stack catches the exception and does not rethrow it, the Integration Server does not log the exception. <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p data-bbox="414 388 495 420">Note:</p> <p data-bbox="414 420 1356 493">The exception may be logged more than once if a service in the call stack, or one of the core classes that the service uses, explicitly logs the exception.</p> </div> <ul style="list-style-type: none"> <li data-bbox="365 525 1385 630">■ Displays, for logged exceptions, the stack trace of the innermost service where the exception first occurred. This stack trace often gets truncated from the log when <code>watt.server.deprecatedExceptionLogging</code> is set to <code>true</code>. <li data-bbox="365 651 1385 697">■ Concatenates the error messages from each of the nested exceptions.

If this parameter is set to `true`, the cause of the exception becomes more difficult to trace. For this reason, Software AG recommends that you do not set this parameter to `true` unless you are executing services that catch exceptions and do not rethrow them.

For more information about interpreting the error log and using the log to help debug services, see [“Debugging Service Exceptions Using the Error Log” on page 947](#).

watt.server.diagnostic.logFiles.maxMB

Specifies the maximum number of megabytes of data that Integration Server reads from the file system for an audit log while collecting diagnostic data. This parameter affects the collection of audit log files only. It does not affect audit log data read from the database, nor does it affect other log files such as the server log, stats log or terracotta-client logs. The default is 250 megabytes. You do not need to restart Integration Server for changes to this parameter to take effect.

watt.server.diagnostic.logperiod

Specifies how many hours of logs are returned when you run the diagnostic tool. The default is 6. When this property is set to 0, the diagnostic utility does not return any log files. It returns only the configurational and run-time data files.

watt.server.diagnostic.port

Specifies the diagnostic port through which you can access Integration Server when it is unresponsive. During installation, Integration Server automatically creates the diagnostic port at 9999. The default is 9999.

Note:

This setting overrides the **Port** setting on the on the Edit Diagnostic Port Configuration screen.

If you are running multiple Integration Servers on the same host machine, the diagnostic port on each server must have a unique port number. Set this parameter for each Integration Server instance to use a unique port number. For more information about the diagnostic port and running multiple Integration Server instances, see [“Adding an HTTP Diagnostic Port” on page 189](#).

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect. To change the diagnostic port without restarting Integration Server, edit the default diagnostic port on the **Security > Ports** page of Integration Server Administrator. For more information, see [“Editing a Port” on page 206](#).

watt.server.diagnostic.tabular

Specifies whether Integration Server should generate diagnostic files in tabular format. If set to `false`, Integration Server generates the files in non-tabular format. The default is `true`.

watt.server.dispatcher.comms.brokerPing

Specifies how often (in milliseconds) triggers (which are Broker clients) should ping the Broker. When there is a firewall between the Integration Server and the Broker, the firewall closes the connection between a trigger and the Broker when the connection becomes idle. To prevent connections from becoming idle, trigger Broker clients periodically ping the webMethods Broker. The default is 1800000 milliseconds (30 minutes).

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.dispatcher.comms.connectionShareLimit

Specifies the number of BrokerClient objects that Integration Server can use to communicate with Broker. Integration Server passes this value to set the following Broker API: `COM.activesw.api.client.BrokerConnectionDescriptor.setConnectionShareLimit` (int value).

For more information, see *webMethods Broker Java Client API Reference*.

Each Broker connection represents a socket open to the Broker. While decreasing the value of the property can increase throughput, it also increases the number of resources and file descriptors Integration Server uses. Set this parameter to either 0 or a value greater than 1. When set to 0, connection sharing is turned off and each Broker client uses its own Broker connection. The default is 5.

Note: Broker does not support a value of 1. If the value is set to 1, Broker issues an exception.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.dispatcher.join.reaperDelay

Specifies how often (in milliseconds) that the Integration Server removes state information for completed and expired joins. The default is 1800000 milliseconds (30 minutes).

watt.server.dispatcher.messageStore.redeliverOriginalMessage

Specifies whether Integration Server stores and redelivers the original message or a modified message for subsequent trigger service execution when retry failure occurs for a trigger service invoked by webMethods messaging trigger. When set to `true`, Integration Server stores and redelivers the message originally sent to the webMethods messaging trigger. When set to `false`, Integration Server stores the message as it is at the point at which retry failure occurs. Integration

Server delivers a modified message to the webMethods messaging trigger for subsequent processing. The default is false.

watt.server.displayDirectories

Specifies whether a browser user can view directories that reside on Integration Server without using Integration Server Administrator. When this parameter is set to true (the default), users can view Integration Server directories. When this parameter is set to false, no directories are displayed.

watt.server.dumpWhitelist

When whitelist class filtering is enabled, indicates whether or not Integration Server generates a log file that lists any classes for which deserialization was attempted but are not part of a `whitelistclasses.xml` file. When `watt.server.dumpWhitelist` is set to true, Integration Server generates a log file that lists any classes that were called but were not in a whitelist classes file. Specifically, Integration Server creates the following file during shut down:

Integration Server_directory /instances/*instanceName*/logs/runtimeClasslist.*time*.xml where *time* is the time in milliseconds since January 1, 1970.

The default value of `watt.server.dumpWhitelist` is false, which indicates that Integration Server does not generate log of classes that were called but were not part of a whitelist classes file.

watt.server.email.charset

Specifies the default character set used for encoding portions that contain non-ASCII characters in email messages. Email fields that contain non-ASCII characters are subject, MIME headers, body text, and attachments. The `pub.client:smtp` service uses this property. The default is `utf-8`. For more information about the `pub.client:smtp` service, see *webMethods Integration Server Built-In Services Reference*.

watt.server.email.from

Specifies the email address the server presents as its From address when sending emails about errors. By default, the server uses `Integration-Server@localhost` for the From Address, where *localhost* is the name of the host on which the Integration Server is running.

watt.server.email.processReplyEmails

Specifies whether Integration Server should process the reply emails that it receives in its email port if the Subject line of the email contains the prefix "Re:". Set the `watt.server.email.processReplyEmails` property to true if you want Integration Server to process reply emails that contain the prefix "Re:" along with a valid service name in the Subject line. If the property is set to false, Integration Server does not process the email if the Subject line contains the prefix "Re:". Instead, Integration Server generates a server log message. The default is false.

watt.server.email.waitForServiceCompletion

Specifies whether Integration Server should wait for the email port to complete the processing of a message before returning the thread used to retrieve the message to the thread pool.

When retrieving and processing messages received by an email port, Integration Server uses one thread to retrieve a message and a second thread to execute the service that processes the message. By default, after using a thread to retrieve a message, Integration Server returns it to the server thread pool, making it available to perform other processing.

If the email port receives a large number of messages in a short period of time, the email port might monopolize the server thread pool for retrieving and processing messages. In addition,

if the service that processes the messages executes slowly, it is possible that the email port message processing will negatively affect the performance of Integration Server.

To avoid this issue, you can limit the number of threads used to process messages received by an email port by setting the `watt.server.email.waitForServiceCompletion` property.

When the property is set to `true`, Integration Server does not return the thread used to retrieve the message to the thread pool until after the service that processes the message executes to completion. As a result, the maximum number of threads used to retrieve and process messages for an email port is equal to 2 or twice the value of the **Number of threads if multithreading is turned on** field.

- If the email port is for a POP3 email server, Integration Server can use up to 2 threads for retrieving and processing messages for the port (one thread to retrieve a message received by the email port and one thread to execute the service that processes the message).
- If the email port is for an IMAP email server and multithreading is disabled, Integration Server can use up to 2 threads for retrieving and processing messages for the port (one thread to retrieve a message received by the email port and one thread to execute the service that processes the message).
- If the email port is for an IMAP email server and multithreading is enabled, Integration Server can use a number of threads equal to twice the value of the **Number of threads if multithreading is turned on** field for retrieving and processing messages for the port. For example, if the value of the **Number of threads if multithreading is turned on** field is 5, Integration Server can use a maximum of 10 threads to retrieve and process messages for the email port (5 threads to concurrently retrieve messages for the email port and another 5 threads to process those messages).

When `watt.server.email.waitForServiceCompletion` is set to `false`, Integration Server does not wait for the service that processes the message to execute to completion before returning the thread used to retrieve the message to the thread pool.

The default is `false`.

Note:

The `watt.server.email.waitForServiceCompletion` parameter applies to all email ports on an Integration Server.

watt.server.enableHotDeployment

Specifies whether hot deployment of packages is enabled on Integration Server. Set to `true` to enable hot deployment. Set to `false` to disable hot deployment. The default is `false`.

The value of `watt.server.enableHotDeployment` is tied to the **Enable** field on the **Settings > Hot Deployment > Edit Hot Deployment Settings** screen in Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to enable or disable hot deployment instead of using the `watt.server.enableHotDeployment` parameter.

watt.server.enterprisegateway.ignoreXForwardedForHeader

Specifies whether Integration Server must ignore the `X-Forwarded-For` request header while processing the rules in Enterprise Gateway Server. If this property is set to `true` then Integration Server ignores the `X-Forwarded-For` request header and considers the proxy server's IP address as the host IP address. If the property is set to `false` then Integration Server obtains the actual host IP address from the `X-Forwarded-For` request header. Default value is `true`.

watt.server.errorMail

Specifies the email address to which Integration Server sends messages about critical server log entries. There is no default.

watt.server.event.audit.async

Specifies whether the event handlers for the audit event are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to audit events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to audit events synchronously. The default is `true`.

watt.server.event.exception.async

Specifies whether the event handlers for the exception event, error event, or journal event are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to exception events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to exception events synchronously. The default is `true`.

watt.server.event.gd.async

Specifies whether the event handlers for guaranteed delivery events (`gdStart` and `gdEnd`) are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to guaranteed delivery events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to guaranteed delivery events synchronously. The default is `true`.

watt.server.event.jmsDeliveryError.async

Specifies whether the event handlers for JMS delivery failure events are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to JMS delivery failure events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to JMS delivery failure events synchronously. The default is `true`.

watt.server.event.jmsRetrievalError.async

Specifies whether the event handlers for JMS retrieval failure events are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to JMS retrieval failure events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to JMS retrieval failure events synchronously. The default is `true`.

watt.server.event.replication.async

Specifies whether the event handlers for replication events are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to replication events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to the replication events synchronously. The default is `true`.

watt.server.event.routing.bodyAsString

When you are using the `pub.event.routing.subscribe` service, you receive an `evt:Event` document as input in the subscription service. This parameter specifies whether you want to map the `evt:body` element inside the `evt:Event` document to the expected document type. When this parameter is set to `false`, Integration Server maps the `evt:body` element inside the `evt:Event` document to the

expected document structure. When this parameter is set to `true`, Integration Server does not map the `evt:body` element inside the `evt:Event` document. The default is `false`.

watt.server.event.routing.runAsUser

Specifies the user who will invoke the service specified in the `pub.event.routing:send` and `pub.event.routing:subscribe` built-in services. Integration Server uses the user specified in this parameter only if no user is specified in the `runAsUser` input parameter of the `pub.event.routing:send` and `pub.event.routing:subscribe` services. The default is `Administrator`.

This user can be defined either internally or in an external directory but must belong to an existing group that has appropriate ACL permissions to invoke the specified service. Integration Server does not validate this user until the service is executed.

The new value of the parameter applies to existing subscriptions as well as new subscriptions created using `pub.event.routing:subscribe`.

watt.server.event.security.async

Specifies whether the event handler for security events is invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to security events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to security events synchronously. The default is `true`.

watt.server.event.session.async

Specifies whether the event handlers for session events (`sessionStart`, `sessionEnd`, and `sessionExpire`) are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to session events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to session events synchronously. The default is `true`.

watt.server.event.stat.async

Specifies whether the event handlers for `stat` (statistics) events are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to the statistics events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to the statistics events synchronously. The default is `true`.

watt.server.event.tx.async

Specifies whether the event handlers for the transaction events (`txStart` and `txEnd`) are invoked asynchronously or synchronously. When this parameter is set to `true`, Integration Server invokes the event handlers (services) that subscribe to transaction events asynchronously. When this parameter is set to `false`, Integration Server invokes the event handlers that subscribe to transaction events synchronously. The default is `true`.

watt.server.eventHandlerCreateSession

Controls whether Integration Server should create a session for the service that is invoked by the Event Manager. When set to `true`, Integration Server creates a session and associates it with the invoked service. When set to `false`, Integration Server does not create a session for the service that is invoked by the Event Manager. The default is `false`.

Note:

When set to `true`, specify a timeout value for the session using the `watt.server.eventHandlerSessionTimeout` parameter.

watt.server.eventHandlerSessionTimeout

When `watt.server.eventHandlerCreateSession` is set to `true`, `watt.server.eventHandlerSessionTimeout` specifies the timeout value for sessions created by Integration Server for services that are invoked by the Event Manager. The default value is 60000 milliseconds.

watt.server.eventHandlerUser

Specifies the user account whose credentials Integration Server uses to execute an event handler service. You can specify a locally defined user account or a user account defined in a central or external directory. Make sure that the user account you specify includes the credentials required by the execute ACLs assigned to the services used as event handlers. The default is Administrator.

watt.server.eventMgr.maxThreads

Specifies the maximum number of threads that you want Integration Server to use for the Event Manager thread pool. The default is 5.

Integration Server verifies that this value is greater than `watt.server.eventMgr.minThreads`. If the value of `watt.server.eventMgr.maxThreads` is less than the value of `watt.server.eventMgr.minThreads`, Integration Server sets the value of `watt.server.eventMgr.maxThreads` to one more than `watt.server.eventMgr.minThreads`. For example, if `watt.server.eventMgr.minThreads` is set to 6 and `watt.server.eventMgr.maxThreads` is set to 5, Integration Server sets the value of `watt.server.eventMgr.maxThreads` to 7.

watt.server.eventMgr.minThreads

Specifies the minimum number of threads that you want Integration Server to use for the Event Manager thread pool. The default is 2.

Integration Server verifies that this value is 1 or greater. If set to a value less than 1, Integration Server resets it to 1.

watt.server.fileEncoding

Specifies the encoding the server is to use when reading and writing text files. This setting has no effect on files stored as Unicode. The default is your JVM's `file.encoding` property.

Note:

To process incoming XML files, use the `watt.server.xml.encoding` parameter. If you have configured Integration Server to use the character encoding specified in the `watt.server.fileEncoding` parameter to process incoming XML files, you must ensure that the value of `watt.server.fileEncoding` parameter is set to the same value specified for `watt.server.xml.encoding`.

watt.server.ftpConnect.message

Specifies the content of the 220 message that Integration Server returns to an FTP client when the client issues a connect request.

When this parameter is set to `true` (the default), Integration Server returns the following messages to the FTP client:

```
Connected to IS_hostname.  
220 IS_hostname FTP server (webMethods Integration  
Server version n.n.n.n) ready.
```

When this parameter is set to `false`, Integration Server returns the following messages to the FTP client:

```
Connected to IS_hostname. 220
```

When this parameter is set to any other value, that value is returned in the 220 message. For example, if you specify Custom FTP connect message, Integration Server will return the following to the FTP client:

```
Connected to IS_hostname. 220 Custom FTP connect message.
```

watt.server.ftp.listingFileAge

Specifies the number of seconds that must elapse before a file that has been updated or created on an Integration Server functioning as an FTP server can be accessed. Files created or updated within the time specified by this parameter will not be part of the results of the FTP LIST command. The default value is 60 seconds.

watt.server.ftp.Login.message

Specifies the content of the 230 message that Integration Server returns to an FTP client when the client issues a login request.

When this parameter is set to `true` (the default), Integration Server returns the following message to the FTP client: 230 User userid logged in.

When this parameter is set to `false`, Integration Server returns the following message to the FTP client: 230

When this parameter is set to any other value, that value is returned in the 230 message. For example, if you specify Custom FTP login message, Integration Server will return the following to the FTP client:

```
230 Custom FTP login message.
```

watt.server.ftp.System.message

Specifies the content of message 215 that Integration Server returns to an FTP client when the client issues a system request. When this parameter is set to `true` (the default), Integration Server returns the following message to the FTP client:

```
215 UNIX Type: L8 Version: webMethods IS FTP <Integration Server n.n.n.n>
```

When this parameter is set to `false`, Integration Server returns the following message to the FTP client:

```
215
```

When this parameter is set to any other value, that value is returned in the 215 message. For example, if you specify Custom FTP system message, Integration Server returns the following to the FTP client:

```
215 Custom FTP system message.
```

watt.server.ftp.usecommandip

Controls whether the `pub.client:ftp` service uses connection information from a NAT server when connecting to an FTP server.

When the `pub.client:ftp` service tries to transfer data to or from an FTP server, Integration Server first connects to the FTP server at the IP address specified by the `pub.client:ftp` service. In response, the FTP server sends back the IP address on the FTP server to which Integration Server should connect to perform the transfer. If the FTP server sits behind a NAT server, the NAT server intercepts this address, translates it, then sends it on to Integration Server.

This property controls whether Integration Server uses the address provided by the NAT server or the address already specified by the `pub.client:ftp` service.

When this parameter is set to `true`, Integration Server bypasses the translated address and tries the address specified by the service. If this attempt fails, Integration Server throws an exception.

When this parameter is set to `false`, the default, Integration Server tries the address provided by the NAT server. If that attempt fails, Integration Server tries the IP address specified on the `pub.client:ftp` service. If both attempts fail, Integration Server throws an exception.

watt.server.getLastError.removeLastError

Integration Server removes the *pipeline/lastError* document from the output of the last invocation of the `pub.flow:getLastError` service if you set the parameter to `true`. When this parameter is set to `false` or not set, the *pipeline/lastError* document is retained. The parameter does not have a default value.

Invoking `pub.flow:getLastError` repeatedly in a flow service increases the pipeline size as the *pipeline/lastError* document has the *lastError* of all previous `pub.flow:getLastError` executions, which leads to "out of memory" issues in Integration Server when the exceptions are high. In such scenarios, you can use the parameter.

Note:

This server configuration parameter is introduced for PIE-72767 in IS_10.5_Core_Fix17.

watt.server.hostAccessMode

Specifies IP access for ports that do not have a custom IP access setting. When this parameter is set to `include`, the server accepts requests from all IP addresses, except those specifically denied on the Integration Server Administrator interface. When this parameter is set to `exclude`, the server denies requests from all IP addresses except those specifically allowed on the Integration Server Administrator interface. The default is `include`.

watt.server.hostAllow

Specifies the name of the host that is allowed service. There is no default.

watt.server.hotDeploymentAutoRecover

Specifies whether or not automatic recover is enabled for hot deployment of packages on Integration Server. Set to `true` to enable automatic recovery of the previous version of a package if hot deployment of the new version of the package fails. Set to `false` to disable automatic recovery of packages if hot deployment fails. The default is `false`.

The value of `watt.server.hotDeploymentAutoRecover` is tied to the **Auto Recover** field on the **Settings > Hot Deployment > Edit Hot Deployment Settings** screen in Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to enable or disable auto recovery for hot deployment instead of using the `watt.server.hotDeploymentAutoRecover` parameter.

watt.server.hotDeploymentTimeout

Specifies the number of seconds that Integration Server waits for in-flight processing to complete before attempting to hot deployment of a packages. If there are any long running tasks that you want to complete before deploying the packages, make sure that you set the timeout value to a value more than the time it takes for the task to complete. The default is 60 seconds.

The value of `watt.server.hotDeploymentTimeout` is tied to the **Timeout** field on the **Settings > Hot Deployment > Edit Hot Deployment Settings** screen in Integration Server Administrator.

Software AG recommends that you use Integration Server Administrator to set the timeout value instead of using the `watt.server.hotDeploymentTimeout` parameter.

watt.server.http.allowOptions

Indicates whether Integration Server allows a client to use the HTTP OPTIONS method to determine the HTTP methods supported by Integration Server. When this parameter is set to `true`, Integration Server allows the OPTIONS method. When this parameter is set to `false` (or any other value), OPTIONS requests from clients receive a "405 Method Not Allowed" response. The default is `true`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.http.authorizationEncoding

Specifies the encoding that the HTTP client (such as a browser) uses when encoding the user name and password for Integration Server. The HTTP client sends the user name and password to Integration Server in the Authorization attribute in the header of the HTTP request. Set the `watt.server.http.authorizationEncoding` property to the encoding that your browser uses for the Authorization attribute. Consult the browser documentation, or contact the browser manufacturer, to find what encoding is used in your environment. The default is UTF-8.

Note:

If two browsers use different encodings for the Authorization attribute, and both send non-ASCII passwords to Integration Server, only the browser whose Authorization encoding matches the value of the `watt.server.http.authorizationEncoding` property will work.

watt.server.http.Content-Security-Policy

Sets the HTTP security header Content-Security-Policy in the responses to requests for accessing the Integration Server Administrator. You can use this property to detect and mitigate attacks such as Cross Site Scripting (XSS) and data injection.

By default, this property is set to "none" and the Content-Security-Policy header is not included in the responses. To secure access to the Integration Server Administrator you can set this property to the value "script-src 'self' 'unsafe-inline'" as follows:

```
watt.server.http.Content-Security-Policy=script-src 'self' 'unsafe-inline'
```

For related information, see `watt.server.http.X-Permitted-Cross-Domain-Policies`, `watt.server.http.Strict-Transport-Security`, `watt.server.http.X-Content-Type-Options`, and `watt.server.http.X-XSS-Protection`.

watt.server.http.header.sameSite

Specifies whether HTTP responses from IS include the SameSite attribute in the Set-Cookie HTTP response header or not, and the value of the SameSite attribute if it is included. Valid values for this parameter are: None, Strict, and Lax. These values and their behavior conform to the HTTP specification.

Note:

- The HTTP standard states that when the SameSite attribute of the Set-Cookie HTTP response header is None, the Secure attribute must be specified. To include the Secure

attribute in the HTTP response header, set the server configuration parameter `watt.server.http.header.useSecure` to `true`.

- If you try to set `watt.server.http.header.sameSite` to `None` in Integration Server Administrator, when the `watt.server.http.header.useSecure` parameter is set to `false`, Integration Server Administrator displays a warning message and does not allow you change the value.
- If you edit the values of the `watt.server.http.header.sameSite` or the `watt.server.http.header.useSecure` parameters directly in the `server.cnf` configuration file and the specified configuration is not valid, Integration Server logs a warning and the invalid configuration is ignored. See [“Setting Up the Server Log” on page 213](#) for more information on Integration Server logging. As the value of `watt.server.http.header.sameSite` is ignored, the `SameSite` attribute is not added to the `Set-Cookie` response header.
- If you set the `Set-Cookie` header and include the `SameSite` attribute using the `pub.flow:setResponseHeader` or `pub.flow:setResponseHeaders` services then Integration Server ignores the value of the `watt.server.http.header.sameSite` parameter.

watt.server.http.header.useHttpOnly

Specifies whether Integration Server includes the `HttpOnly` attribute in the `Set-Cookie` header in the response to an HTTP client. When this property is set to `true` (the default), Integration Server includes the `HttpOnly` attribute in the `Set-Cookie` header of the HTTP response to an HTTP client. Software AG recommends setting this property to `true` because it prevents a client side script from accessing a protected cookie. If you are working with an HTTP client that does not support the `HttpOnly` attribute, set this property to `false`.

watt.server.http.header.useSecure

Specifies whether Integration Server includes the `secure` attribute in the `Set-Cookie` header in the response to an HTTP/S client. When the `watt.server.http.header.useSecure` property is set to `true` (the default), Integration Server includes the `secure` attribute in the `Set-Cookie` header of the HTTPS response to an HTTP/S client. Software AG recommends setting this property to `true` because it ensures that the cookie is always encrypted while being transmitted from client to server.

Important:

The value of the `watt.server.http.header.sameSite` parameter impacts how Integration Server uses this parameter. See `watt.server.http.header.sameSite` for more information.

watt.server.http.interceptor.enabled

Specifies whether the inbound HTTP interceptor is enabled for Integration Server. When set to `true`, Integration Server uses the inbound HTTP interceptor implementation identified by the `watt.server.http.interceptor.impl` property. The default value of this property is `false`.

Note:

The first time you set this property to `true`, you must restart Integration Server for changes to the property to take effect.

watt.server.http.interceptor.impl

Specifies the fully qualified name of the Java class that implements the inbound HTTP interceptor. The provided class must implement the `com.softwareag.is.interceptor.HttpInterceptorIFC` interface and provide implementations of both methods in the interface (`preProcess` and `postProcess`). The compiled class must be available on the server classpath.

Important:

You must restart Integration Server for a change to this parameter to take effect.

watt.server.http.interceptor.outbound.enabled

Specifies whether an outbound HTTP interceptor is enabled for Integration Server. When set to true, Integration Server uses the outbound HTTP interceptor implementation identified by the `watt.server.http.interceptor.outbound.impl` property. The default value of this property is false.

Important:

The first time you set this property to true, you must restart Integration Server for changes to the property to take effect.

watt.server.http.interceptor.outbound.impl

Specifies the fully qualified name of the Java class that implements the outbound HTTP interceptor. The provided class must implement the `com.softwareag.is.interceptor.HttpOutboundInterceptorIFC` interface and provide implementations of both methods in the interface (`preProcess` and `postProcess`). The compiled class must be available on the server classpath.

Important:

You must restart Integration Server for a change to this parameter to take effect.

watt.server.http.interceptor.preprocess.sizeLimit

Specifies the maximum number of bytes that an inbound HTTP interceptor reads during preprocessing. If the `HttpRequest`'s input exceeds this value, the inbound HTTP interceptor reads up to this limit and then ignores the rest of the input stream. Set this property to `N [KB|MB|GB]`, where `N` is any valid integer. Do not include any spaces between the integer and the unit of measure. If you do not specify a unit of measure, Integration Server treats the supplied `N` value as bytes. The default value is `-1`, which indicates there is no limit to the size of the request bytes that will be passed to the inbound HTTP Interceptor: The inbound HTTP interceptor will read the entire input stream.

watt.server.http.jsonFormat

Specifies how Integration Server handles JSON content in HTTP client requests. When this property is set to `parsed` (the default), Integration Server parses JSON content into pipeline variables automatically. Clients can then use JSON to compose pipelines and execute existing services without having to call `pub.json:jsonStreamToDocument`.

If `watt.server.http.jsonFormat` is set to `stream`, Integration Server places a `jsonStream` variable containing the JSON content in the pipeline as an `InputStream`. You can then use `pub.json:jsonStreamToDocument` to parse `jsonStream` into pipeline variables.

If `watt.server.http.jsonFormat` is set to `bytes`, Integration Server places the incoming stream of JSON into a byte array.

You can override the behavior specified by `watt.server.http.jsonFormat` in individual requests by adding the `jsonFormat` query parameter to the request URI. For example, if `watt.server.http.jsonFormat` is set to `parsed`, a client can override this setting and instead specify a value of `stream` for the request by entering a URI as follows:

```
/invoke/myfolder/myservice?jsonFormat=stream
```

This will result in the request being placed in a *jsonStream* variable in the pipeline as an *InputStream* containing the JSON content.

watt.server.http.listRequestVars

Specifies how Integration Server handles duplicate query tokens in an HTTP request.

When set to	Integration Server
always	<p>Converts duplicate and non-duplicate tokens into lists. For example:</p> <pre>http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z</pre> <p>becomes</p> <pre>INVOKE folder:service {arg1=X, arg1List=[X,Y], arg2=Z, arg2List=[Z]}</pre>
asNeeded	<p>Converts duplicate tokens into Lists. Non-duplicate tokens are not converted. For example:</p> <pre>http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z</pre> <p>becomes</p> <pre>INVOKE folder:service {arg1=X, arg1List=[X,Y], arg2=Z}</pre> <p>This is the default setting.</p>
error	<p>Throws a <i>ServiceException</i> if duplicate tokens are found in the URL query. Non-duplicate tokens are not converted. For example:</p> <pre>http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z</pre>
never	<p>Ignores duplicate tokens found in the URL query. For example:</p> <pre>http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z</pre> <p>becomes</p> <pre>INVOKE folder:service {arg1=X, arg2=Z}</pre>

watt.server.http.preserveUriReservedChars

Specifies whether Integration Server should decode percent-encoded URI paths in requests before evaluating URI paths. When this property is set to *true* (the default), Integration Server defers decoding percent-encoded reserved characters including white space, until after evaluating URI paths. When set to *false*, Integration Server decodes percent-encoded reserved characters including white space prior to evaluating the URI paths.

Note:

The value of *watt.server.http.preserveUriReservedChars* is significant only for the REST requests that use the *rest* directive. For requests using the *rest* directive, the last tokens in the path identify the instance of a resource or provide application-specific information. These tokens become the *\$resourceID* and *\$path* variables in the pipeline. This portion of the URI path can contain percent-encoded reserved characters with white space. When

`watt.server.http.preserveUriReservedChars` is set to `true`, Integration Server decodes the percent-encoded characters after evaluating the URI path, but before invoking the service.

For requests that use the `invoke` directive, the entire path of the URI after `/invoke` refers to folders and services in the Integration Server namespace. Folders and services cannot contain any URI-reserved characters including white space, therefore any request that uses the `invoke` directive and contains a percent-encoded reserved character along with white space in its path will never identify an actual Integration Server service and will always result in an HTTP 404 response.

watt.server.http.reauth.user-agent.list

Specifies a semicolon-delimited list of HTTP clients from which you want Integration Server to request re-authentication after an HTTP session expires. By default, this parameter is set to `Firefox;MSIE` so that requests from the Mozilla Firefox and Microsoft Internet Explorer browsers will be asked to re-authenticate if the associated HTTP session has expired. For user agents not specified on this list, Integration Server sends a new session cookie when the session expires.

If you remove this property and restart Integration Server, the value of this property resets to `Firefox;MSIE`.

watt.server.http.request.supportCompression

Specifies whether Integration Server needs to support HTTP request compression. By default, this parameter is set as `false`. If you want Integration Server to support decompression of the payload after receiving a compressed payload in HTTP request, set this parameter as `true`. For more information on HTTP request compression, see *webMethods Integration Server Built-In Services Reference* guide.

Integration Server decompresses the payload based on the `Content-Encoding` request header defined in the HTTP request. The request header defines the compression scheme used to compress the payload. Integration Server uses the same compression scheme to decompress the payload. The supported compression schemes are *Gzip* and *Deflate*.

If the parameter is set as `false` and Integration Server receives compressed payload in HTTP request then it ignores the request.

watt.server.http.response.supportCompression

Specifies whether Integration Server needs to support HTTP response compression. By default, this parameter is set as `false`. If you want Integration Server to support compression of the payload before sending the HTTP response, set this parameter as `true`. For more information on HTTP response compression, see *webMethods Integration Server Built-In Services Reference* guide.

Integration Server compresses the payload based on the `Accept-Encoding` request header defined in the HTTP request. The request header defines the suggested compression scheme that Integration Server can use to compress the payload. The supported compression schemes are *Gzip* and *Deflate*.

If the parameter is set as `false` and Integration Server receives an HTTP request to compress the response payload then it sends an uncompressed response.

watt.server.http.returnValueException

Specifies whether Integration Server should return a stack trace in the HTTP/SOAP response it sends to the invoking client when an invoked service ends in error. You can specify one of the following:

Value	Tells Integration Server
true	To return a stack trace in the HTTP/SOAP response. This is the default.
false	Not to return a stack trace.
webMethods	Return a stack trace only if the client's HTTP user agent is set to webMethods or has the same value as the user agent specified on the watt.net.userAgent parameter.

watt.server.http.securityRealm

Specifies the name of the Integration Server security realm. This value is included in the headers of HTTP responses that Integration Server sends to clients that request authentication. The default value is **Integration Server**.

watt.server.http.Strict-Transport-Security

Specifies whether Integration Server includes Strict-Transport-Security header in response headers. Value specified for this parameter is set in the response headers of web requests to access the Integration Server Administrator. Users are free to set any valid value to this parameter. If the value is set to none or left blank, then the Strict-Transport-Security header is not included in the response headers. The default is none.

Example: `watt.server.http.Strict-Transport-Security=max-age=300 ; includeSubDomains`

For more information about valid values for this parameter, see the HTTP Strict Transport Security (HSTS) section of *Internet Engineering Task Force (IETF) November 2012 specification*.

Note:

The HTTP Strict-Transport-Security response header (HSTS) allows web servers to inform that web browsers should only access them using the secure HTTPS connections and not using HTTP connections.

watt.server.http.uriPath.decodePlus

Specifies whether or not the '+' character is interpreted as a ' ' when it appears in the path component of the request URI. When set to true, the default, Integration Server converts '+' to ' ' when processing the request URI. When set to false, Integration Server does not convert '+' to ' ' when processing the request URI. This parameter affects only the handling of '+' characters in the request URI's path component. Any '+' characters that appear in the request URI's query will be converted to ' ' characters regardless of the value of `watt.server.http.uriPath.decodePlus`.

watt.server.http.useAcceptHeader

Specifies whether Integration Server supports the Accept header field in an HTTP request. The Accept header field specifies a content type or types the HTTP client will accept in the HTTP response. The default setting is true. This parameter must be set to true if you want to use the `watt.server.content.type.mappings` parameter to map wildcard content types in the Accept header field to specific content types.

watt.server.http.X-Content-Type-Options

Specifies whether Integration Server includes the X-Content-Type-Options header in response headers. Value specified for this parameter is set in the response headers of web requests to access the Integration Server Administrator. Default value is None. You can set any valid value

for this parameter. If the value is set to *None* or left blank, then the X-Content-Type-Options header is not included in the response headers.

Example: `watt.server.http.X-Content-Type-Options=nosniff`

For more information about valid values for this parameter, see the X-Content-Type-Options section of *Fetch 31 January 2018 specification*.

Note: Integration Server includes the X-Content-Type-Options header in the response header to indicate that the MIME types advertised in the Content-Type headers are followed.

watt.server.http.x-frame-options

Controls how Integration Server is to handle the X-Frame-Options attribute in response headers.

Note: Integration Server includes the X-Frame-Options attribute in the response header to requests for webpages, as defined in the HTTP Header Field X-Frame-Options specifications. X-Frame-Options is not included in responses to requests for service invocation, such as those including the `invoke`, `rest`, `restv2`, or `soap` directives. It is only included in responses to requests for webpages, such as `https://my-server/MyPackage/my-page.html`.

Set this parameter to one of the following values:

Specify	To indicate that
SAMEORIGIN	The client's browser is to allow Integration Server pages to be displayed in an HTML frame only if the frame is on a page from the same server. This is the default value.
ALLOW-FROM <i>other_origin</i>	The client's browser is to allow Integration Server pages to be displayed in an HTML frame only if the frame is on a page from the same server specified by <i>other_origin</i> .

Note:

The value DENY is defined for the X-Frame-Options attribute but is not allowed for Integration Server. DENY means that the page can never appear in a frame, regardless of the frame's origin. This would cause Integration Server Administrator to be unusable. If `watt.server.http.x-frame-options` is set to DENY, that value is ignored and SAMEORIGIN is used instead.

If you do not want Integration Server to include the X-Frame-Options attribute in response headers, remove the value from the `watt.server.http.x-frame-options` property. For example: `watt.server.http.x-frame-options=` The property can be set on the **Settings > Extended** page of Integration Server Administrator. Changes to this property take effect immediately; the server does not need to be restarted.

watt.server.http.X-Permitted-Cross-Domain-Policies

Sets the HTTP security header X-Permitted-Cross-Domain-Policies, which informs clients what cross-domain policies they can use for accessing the Integration Server Administrator. By default, this property is empty and the X-Permitted-Cross-Domain-Policies header is not included in the responses. To secure access to the Integration Server Administrator you can set this property to "none", which stops clients from loading data from your domain.

For related information, see `watt.server.http.Content-Security-Policy`, `watt.server.http.Strict-Transport-Security`, `watt.server.http.X-Content-Type-Options`, and `watt.server.http.X-XSS-Protection`.

watt.server.http.X-XSS-Protection

Specifies whether Integration Server includes X-XSS-Protection in the response headers. Value specified for this parameter is set in the response headers of web requests to access Integration Server Administrator. Default value is `None`. You can set any valid value for this parameter. If the value is set to `None` or left blank, then the response header does not include the X-XSS-Protection.

Example: `watt.server.http.X-XSS-Protection=1; mode=block`

For more information about valid values for this parameter, refer to your browser documentation.

Note:

The HTTP X-XSS-Protection response headers stops web pages from loading when they detect reflected cross-site scripting (XSS) attacks.

watt.server.http.xmlFormat

Specifies whether Integration Server parses incoming XML documents automatically or passes them directly to the service. Use this property to set the default behavior for how Integration Server should pass XML documents via HTTP. Specify one of the following values:

Specify	To indicate that
bytes	Integration Server passes XML bytes directly to the requested service without parsing. When this parameter is set to <code>bytes</code> , the XML appears in the input pipeline of the service as byte array, named <code>xmlBytes</code> .
enhanced	Integration Server parses the XML automatically using the enhanced XML parser and passes it to the service as an enhanced node object named <code>node</code> that implements the <code>org.w3c.dom.Node</code> interface.
node	Integration Server parses the XML automatically using the legacy XML parser and passes it to the service in a parameter named <code>node</code> of type <code>com.wm.lang.xml.Node</code> . This is the default behavior.
stream	Integration Server passes XML streams directly to the requested service without parsing. When this parameter is set to <code>stream</code> , the XML appears in the input pipeline of the service as an <code>InputStream</code> , named <code>xmlStream</code> .

If a value for `watt.server.http.xmlFormat` is not specified or is invalid, the default XML parsing behavior is "node".

This feature can be used when submitting and receiving XML via HTTP or HTTPS only.

Note:

The default established by this parameter can be overridden in any individual client request that uses the `xmlFormat` argument in the URL. For more information about the `xmlFormat` argument, see *webMethods Service Development Help*.

Note:

Changing the default `xmlFormat` for all of Integration Server might cause existing services and applications to fail. Consider changing the `xmlFormat` on a service or application basis instead by using the **Default `xmlFormat`** property for a service. For more information about the **Default `xmlFormat`** property, see *webMethods Service Development Help*.

watt.server.httplog

Specifies whether Integration Server should log outbound HTTP/HTTPS requests to a separate log file. If set to `true`, Integration Server creates the *Integration Server_directory* \instances*instance_name*\logs\http.log file, which logs data for HTTP/HTTPS requests. The http.log file is created separately from the data captured in the server log. The default is `false`. Integration Server logs the following in http.log:

Parameter	Description
CURRENT-TIME	Time of the host Integration Server at the time the request is made.
SIZE-OF-RESPONSE	Size of the data (in bytes) in the response. This is either the current size of the in-memory byte buffer or the size of the input stream.
TIME-TAKEN	Time (in milliseconds) it takes the Integration Server to process the request. This parameter does not include transport time.
HOST-ADDRESS	IP address of the host Integration Server.
REQUESTED-URL	The URL requested by the client.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.hostDeny

Specifies the name of the host that is denied service. There is no default.

watt.server.idr.reaperInterval

Specifies the interval, measured in minutes, at which the scheduled system service Message History Sweeper executes and removes expired document history entries. The default is 10 minutes.

watt.server.illegalNSChars

Specifies the characters that you cannot use when naming a package, folder or service. The default is `?`-#&@^!%*:$. \ / ' ; , ~ +=) (| } { [> < "`.

watt.server.illegalUserChars

Specifies characters that you cannot use in user names. The default is `"\<&`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Important:

Do not remove characters from this property. If you remove characters and create users that contain those characters, Integration Server cannot parse users.cnf and will be unable to start.

watt.server.inetAddress

Specifies the IP address on which the server is to listen for incoming requests. By default, Integration Server listens on all available IP addresses. To limit the server to listening on a single IP address, specify the IP address as the value of this property.

watt.server.invokeDirective

Specifies an alternative word to use for the invoke directive in URLs that invoke services on Integration Server. By default, this parameter is set as `watt.server.invokeDirective=invoke`, which means users must specify the invoke directive as `invoke` (`http://host:port/invoke/folder/service_name`). To allow users to specify the invoke directive as either `invoke` or an alternative word, set this parameter to the alternative word. For example, to allow users to specify the invoke directive as either `invoke` or `submit`, (`http://host:port/invoke/folder/service_name` or `http://host:port/submit/folder/service_name`), set this parameter as `watt.server.invokeDirective=submit`.

watt.server.invoke.maxRetryPeriod

Specifies the total amount of waiting time (in milliseconds) that can elapse if the Integration Server makes the maximum attempts to retry a service. The default is 15,000 milliseconds (15 seconds). When configuring retries for an individual service, the value calculated by multiplying **Max attempts** value by the **Retry interval** cannot exceed the value set by this server parameter. For more information about configuring service retry, see *webMethods Service Development Help*.

watt.server.java.unicode

Specifies whether the source code for Java services is stored in Unicode encoding. The default is `false`. Set this value to `true` if the source code contains characters that cannot be rendered in the server's native encoding.

watt.server.jca.connectionPool.createConnection.interrupt.waitTime

Specifies the wait time, measured in milliseconds, that elapses before Integration Server interrupts a connection creation thread that is in a wait state. There is no default value.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jca.connectionPool.threadInterrupt.waitTime

Specifies the maximum number of milliseconds that a thread can take while creating or closing a connection before the pool interrupter thread interrupts the thread. After the specified time elapses, the pool interrupter thread considers the thread to be blocked and interrupts it. There is no default value. The pool interrupter thread, which is a monitoring thread for the connection pool, only executes if this server configuration property has a value that is greater than zero.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jca.connectionPool.threadInterrupter.sleepTime

Specifies the number of milliseconds the pool interrupter thread sleeps between sweeps for server threads that became blocked while creating or closing a connection. When the sleep time elapses, the pool interrupter thread, which is a monitor thread for the connection pool, checks for server threads that became blocked while attempting to create or close a connection from the connection pool. The default is 2000 milliseconds.

watt.server.jca.connectionPool.thresholdWaitingRequest

When enabled, this property represents the percentage value that is used in addition to the configured maximum number of connections (set by the Maximum Pool Size parameter on the Connections page) for the connection pool. For example, setting the property as `watt.server.jca.connectionPool.thresholdWaitingRequest=20` sets the threshold to 120% of configured maximum number of connections.

If the property is not defined or if the value is less than or equal to zero, the feature remains disabled.

When this property is enabled, the connection pool ensures that the waiting connection requests plus the busy connections in the connection pool do not exceed the threshold limit.

watt.server.jca.transaction.rollbackOnWriteFailure

If Integration Server cannot store the status of a transaction and its participating resources in the XA recovery store (for example, because the store is corrupted), specifies whether Integration Server should try to continue with the transaction anyway (`false`) or try to roll it back (`true`). Setting the parameter to `false` involves some risk; if Integration Server ends abnormally, no statuses will have been saved to the XA recovery store, and Integration Server will not be able to resolve the uncompleted transaction or give you the chance to resolve it manually.

watt.server.jca.transaction.writeRecoveryRecord

Specifies whether Integration Server maintains XA transaction information for use with XA transaction recovery. If Integration Server does not save XA transaction information, uncompleted XA transactions cannot be recovered using Integration Server. That is, Integration Server does not attempt to recover incomplete XA transactions automatically and you cannot use Integration Server Administrator to manually recover or resolve an incomplete transaction. Specify `true` to enable XA transaction recovery. Specify `false` to disable XA transaction recovery. The default is `true`.

watt.server.jdbc.datadirect.snoop.default

Specifies the default settings for the DataDirect Snoop tool for DataDirect Connect JDBC drivers. The DataDirect Snoop tool logs network packets between Integration Server and an external RDBMS. The resulting log file can be used for tracing and diagnostic purposes.

Note:

This option is for use with an external RDBMS only. It is not for use with the embedded IS internal database.

Set this parameter on each DataDirect Connect JDBC driver for which you want to log data.

Important:

To enable this feature for use with Integration Server, you must select the **Snoop** option on either the **Edit JDBC Connection Pool Alias** screen or the **Create JDBC Connection Pool Alias** screen of the Integration Server Administrator.

The default value for this parameter is:

```
ddtdbg.ProtocolTraceEnable=true;ddtdbg.ProtocolTraceMaxLine=16; ddtdbg.ProtocolTraceLocation=/logs/s
```

Where *<alias_name>* is the name of the JDBC connection pool alias.

Note:

For DB2, include the following command at the end of the value:

```
ddtdbg.ProtocolTraceEBCDIC=true
```

The default value defines the name and location of the log file and DataDirect Snoop tool attributes. Typically, you do not need to change the default value. However, you can modify the value if the attributes do not meet the needs of your system. If you change the log file location, be aware that the diagnostic tool collects data from the *Integration Server_directory/instances/instance_name/logs/snoop* directory. If you change the log file location, the diagnostic utility might not import the data logged by the DataDirect Snoop tool. For more information about using the diagnostic utility, see [“Using the Diagnostic Utility” on page 939](#). For more information about setting the DataDirect Snoop tool attributes, consult the documentation on the DataDirect website.

Important:

If you change the value of this parameter, you must restart the connection pool for the changes to take effect.

watt.server.jdbc.datadirect.spy.default

Specifies the default settings for the DataDirect Spy diagnostic feature for DataDirect Connect JDBC drivers. DataDirect Spy logs JDBC calls and SQL statement interactions between Integration Server and an external RDBMS. The resulting log file can be used for tracing and diagnostic purposes.

Note:

This option is for use with an external RDBMS only. It is not for use with the embedded IS internal database.

Set this parameter for each DataDirect Connect JDBC driver for which you want to log data.

Important:

To enable this feature for use with Integration Server, you must select the **Spy** option on either the **Edit JDBC Connection Pool Alias** screen or the **Create JDBC Connection Pool Alias** screen of the Integration Server Administrator.

The default value for this parameter is:

```
SpyAttributes=(log=(file)/logs/spy/<alias_name>.log;logTName=yes;timestamp=yes)
```

Where *<alias_name>* is the name of the JDBC connection pool alias.

The default value defines the name and location of the log file and DataDirect Spy attributes. Typically, you do not need to change the default value. However, you can modify the value if the attributes do not meet the needs of your system. If you change the log file location, be aware that the diagnostic tool collects data from the *Integration Server_directory/instances/instance_name/logs/spy* directory. If you change the log file location, the diagnostic utility might not import the data logged by DataDirect Spy. For more information about the

diagnostic utility, see [“Using the Diagnostic Utility” on page 939](#). For more information about setting DataDirect Spy attributes, consult the documentation on the DataDirect website.

Important:

If you change the setting of this parameter, you must restart the connection pool for the changes to take effect.

watt.server.jdbc.defaultDriver

For use with WmDB. Specifies the name of the Java class for the driver you want to use to connect to databases when no driver name is supplied for a database alias.

watt.server.jdbc.derby.commitTolerance

Specifies the length of time, in milliseconds, that Integration Server waits for a commit to the embedded database to complete before removing the associated connection from the connection pool. Integration Server uses this property when it is configured to use the embedded database for any of the ISInternal, DocumentHistory, and Xref functions. If the commit process takes longer than the value defined by this property, Integration Server removes the associated connection from the connection pool so that a new connection can be initiated for future requests. The default value for this property is 150 milliseconds.

watt.server.jdbc.driverList

Specifies a comma-delimited list of JDBC drivers you want the server to load when it initializes. There is no default.

Note:

This parameter is for use with the WmDB package only.

watt.server.jdbcLogFile

Specifies the name of the log file that contains JDBC log activity. When the `watt.server.jdbcLogging` property is enabled (set to `on`), the server logs activity to this file. The default value is `jdbc.log`. It is located in the *Integration Server_directory* \instances\instance_name\logs folder. Note that the log file is generated after WmDB makes a connection to the database.

Note:

This parameter is for use with the WmDB package only.

watt.server.jdbc.loginTimeout

Sets the maximum time, in seconds, that the JDBC driver on Integration Server is to wait for a response while attempting to connect to a database. If Integration Server does not receive a response in the allotted time, it terminates the request, logs the error and moves on. The default value is 60 seconds.

If Integration Server is unable to connect to the audit logging database within the number of seconds specified by `watt.server.jdbc.loginTimeout`, Integration Server removes all connections from the ISCoreAudit JDBC connection pool and recreates them. If the connection problem is resolved quickly (such as a momentary network interruption), Integration Server reconnects to the audit logging database. If Integration Server is unable to connect to the audit logging database, such as when the database is offline, Integration Server writes an exception to the server log.

Important:

If you change the setting of this parameter, you must restart the ISCoreAudit JDBC connection pool for the changes to take effect. To restart the pool, in Integration Server Administrator, go to the **Settings > JDBC Pools** page and click **Restart** for the ISCoreAudit functional alias.

watt.server.jdbcLogging

Specifies whether Integration Server is to enable logging on `java.sql.DriverManager` in order to log JDBC activity. When set to `on`, the server writes the log file data to the file specified by `watt.server.jdbcLogFile`. The default is `off`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

This parameter is for use with the WmDB package only.

watt.server.jdbc.moreResults

Specifies if the Integration Server is to process the stored procedure results from a single result set or multiple result sets. When set to `true`, Integration Server processes and returns information produced by multiple result sets. When set to `false`, Integration Server processes information produced from a single result set. The default is `false`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

This parameter is for use with the WmDB package only.

watt.server.jdbc.sp.mandateParams

Specifies whether the `$data` input parameter of the `pub.db:call` service expects values to invoke a stored procedure through the `pub.db:call` service. Set this property to `false` if you do not want to specify values for the `$data` input parameter. Set this property to `true` if you want the `pub.db:call` service to expect values for the `$data` input parameter. The default is `true`.

watt.server.jdbc.statementCache

Specifies whether Integration Server is to cache prepared statements for later use. When set to `true`, the server saves prepared statements in a local cache. When the server receives subsequent requests to a database, it can reuse the prepared statements in the cache instead of recreating them each time a request is made. To use Java heap space efficiently, the server removes cached statements automatically if it is not reused within 30 seconds. The default is `false`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

This parameter is for use with the WmDB package only.

watt.server.jms.csq.maxRedeliveryCount

Specifies the number of attempts to redeliver a document from the client side queue. Specify a positive integer value. The default is 5. If Integration Server is unable to deliver the document in the configured number of attempts, the document is dropped from the client side queue and not delivered. In this situation, Integration Server raises a `JMSDeliveryFailureEvent`.

Note:

Specifying a large value for this property can impact Integration Server performance if the failure to deliver the document is not a transient error, such as the Broker being unavailable. If the reason a document cannot be delivered is because there is problem with the document itself, Integration Server performance is impacted because it will continue to attempt to redeliver the document for the number of times you configure.

Note:

This parameter applies to JMS messaging, regardless of the JMS provider. The comparable property for webMethods messaging where Broker is the messaging provider is `watt.server.publish.maxCSQRedeliveryCount`. The comparable parameter for webMethods messaging where Universal Messaging is the messaging provider is `watt.server.messaging.csq.maxRedeliveryCount`.

watt.server.jms.csq.publishDelayWhileDraining

Specifies the number of seconds that Integration Server waits before publishing a JMS message to the client side queue (CSQ) while the CSQ drains. To use the delay when the CSQ contains one or more messages, specify an integer greater than 0 but less than or equal to 60.

Specify 0 to instruct Integration Server to vary the length of the delay depending on the number of messages in the CSQ. When `watt.server.jms.csq.publishDelayWhileDraining` is set to 0, Integration Server waits for:

- 6 seconds when the number of messages in the CSQ is greater than 0 but less than or equal to 1000
- 8 seconds when the number of messages in the CSQ is greater 1000 but less than or equal to 5000
- 10 seconds when the number of messages in the CSQ is greater than 5000

`watt.server.jms.csq.publishDelayWhileDraining` is 0.

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.connection.monitorPeriod

Specifies the frequency (measured in seconds) with which Integration Server checks the state of a connection to the JMS provider. The minimum is 1 second. If this parameter is set to less than 1, Integration Server uses the default value instead. The default is 45 seconds.

watt.server.jms.connection.pingDestination

Configures Integration Server to send a keep-alive message to a destination on the JMS provider, thereby preventing the firewall from terminating the connection between Integration Server and a third-party JMS provider. A keep-alive message is a blank, non-persistent JMS message with an immediate time out.

Set this parameter to one of the following values:

Specify	To
blank	Attempt to create a JMS session using the connection between Integration Server and the JMS provider. This is the default.
temp	Create a temporary queue on the JMS provider and send a keep-alive message to the temporary queue. Temp is not case-sensitive.
<destinationName>	Send a keep-alive message to the specified destination on the JMS provider. If the specified destination does not exist on the JMS provider, Integration Server periodically attempts to create a JMS session using the connection between Integration Server and the JMS provider.

Note:

The frequency with which Integration Server sends a keep-alive message is determined by the value of `watt.server.jms.connection.pingPeriod`.

watt.server.jms.connection.pingPeriod

Specifies how often, measured in seconds, Integration Server should ping the JMS provider. The ping serves as a keep alive request sent to the JMS provider. The default is 300 seconds (5 minutes).

watt.server.jms.connection.retryPeriod

Specifies the length of time, in seconds, that Integration Server waits between connection attempts when a connection to the JMS provider or JNDI provider fails. The default is 20 seconds.

watt.server.jms.connection.update.blockingTime

Specifies the maximum amount of time, measured in milliseconds, that a `pub.jms*` service will wait for a connection while the connection used by the service is being updated. If Integration Server does not restart the connection before the blocking time elapses, Integration Server throws an `ISRuntimeException` and the sending service fails. A value of 0 (zero) indicates that Integration Server does not block the `pub.jms*` services while the JMS connection alias updates; Integration Server throws an `ISRuntimeException` immediately. The maximum value is 10,000 milliseconds (10 seconds). The default is 1000 milliseconds.

Integration Server resets the `watt.server.jms.connection.update.blockingTime` parameter to the default value if an invalid value is entered.

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.connection.update.restartDelay

Specifies the length of time, measured in milliseconds, Integration Server waits for services sending JMS messages to execute to completion when the connection used by the services needs to be updated. After the restart delay elapses, Integration Server refreshes the connection using the updated connection factory object and then restarts the connection. If the restart delay elapses before the `pub.jms*` services execute to completion, Integration Server throws an `ISRuntimeException` and the sending service fails. The maximum value is 10,000 milliseconds (10 seconds). The default is 500 milliseconds.

Integration Server resets the `watt.server.jms.connection.update.restartDelay` parameter to the default value if an invalid value is entered.

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.csq.batchProcessingSize

Specifies the maximum number of messages Integration Server retrieves from the client side queue and then sends to the JMS provider. Integration Server places sent messages in the client side queue when the connection to the JMS provider fails. Integration Server begins to drain the client side queue after the connection to the JMS provider becomes available. The default is 25.

watt.server.jms.guaranteedMultisend.alwaysUseTxLogging

Specifies whether or not Integration Server always uses XA transaction logging when sending a JMS message in accordance with a multisend guaranteed policy. In XA transaction logging, Integration Server logs the state of each transaction. The XA transaction logging allows Integration Server to recover any transactions that did not complete due to Integration Server failure. While this is the most reliable way to ensure the integrity of a transaction, it may be expensive in terms of performance and may not always be necessary. When this property is set to true, Integration Server always uses XA transaction logging and can perform XA transaction recovery for a multisend guaranteed policy regardless of the connection transaction type. When set to false, Integration Server uses XA transaction logging only if the connection transaction type is `XA_TRANSACTION`. The default is false.

watt.server.jms.nirvana.durableSubscriber.includeClientId

Specifies whether the client ID of the JMS connection alias will be used as the prefix in the names of durable subscribers created using Designer. Set to true to use the value of the **Client ID** property in the JMS connection alias as a prefix in the durable subscriber name. Even if the `watt.server.jms.nirvana.durableSubscriber.includeClientId` is set to true, if the **Client ID** property is empty the durable subscriber name will not have a client ID prefix. Set to false to omit the use of the client ID as a prefix. The default is true.

watt.server.jms.producer.pooledSession.timeout

Specifies the number of milliseconds for which an inactive entry remains in the default session pool or a destination-specific pool before Integration Server removes the entry. Integration Server uses this value when -1 is specified for the **Idle Timeout** field for a JMS connection alias. The default is 60000.

watt.server.jms.trigger.concurrent.consecutiveMessageThreshold

Specifies the consecutive number of successful requests for more messages that a concurrent JMS trigger must make before the trigger becomes multithreaded. For example, when this property is set to 1000, Integration Server begins using multiple threads for a concurrent JMS trigger when the JMS trigger retrieves messages from the JMS provider in 1000 consecutive requests. The default for this property is 0, which indicates that Integration Server does not use a threshold to determine when JMS trigger become multithreaded. When threshold is not used, the concurrent JMS trigger will always be multithreaded when more than one message is available for processing. For more information about using a threshold for thread usage for a concurrent JMS trigger, see [“Establishing a Threshold for Using Multiple Threads with a Concurrent JMS Trigger”](#) on page 868.

Note:

The threshold for using multiple threads with concurrent JMS triggers applies to all the concurrent JMS triggers on the Integration Server. This includes standard JMS triggers, SOAP-JMS triggers, and WS-Endpoint triggers.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jms.trigger.concurrent.primaryThread.pollingInterval

Specifies the length of time (in milliseconds) for which the primary server thread for a concurrent JMS trigger polls the JMS provider for more messages. Each concurrent JMS trigger has a server thread that retrieves messages from the JMS provider. This thread is considered to be the primary thread for the concurrent JMS trigger. A short polling interval increases the amount of CPU used by Integration Server and increases the load on the JMS provider. However, a short polling interval can improve performance under heavy load and in request-reply scenarios. The default is 500 milliseconds.

If a value of 0 is specified, Integration Server uses the `javax.jms.MessageConsumer.receiveNoWait()` API to retrieve messages from the JMS provider. The `javax.jms.MessageConsumer.receiveNoWait()` API may or may not be supported by all JMS providers. If the JMS provider from which a JMS trigger receives messages does not support this API then the JMS trigger will be disabled. If this happens, modify the value of `watt.server.jms.trigger.concurrent.primaryThread.pollingInterval` to be a positive integer. and enable the JMS trigger.

Note:

The default value of 500 milliseconds is not optimal when connecting to Universal Messaging. To improve performance when Universal Messaging is the JMS provider, Integration Server uses a minimum primary polling interval of 3000 milliseconds. If the `watt.server.jms.trigger.concurrent.primaryThread.pollingInterval` property is set to a value greater than 3000, Integration Server uses the property value for connections to Universal Messaging.

Note:

A longer polling interval may affect the performance of JMS triggers with joins because the JMS trigger must retrieve messages from multiple destinations. If one of the destinations from which a JMS trigger retrieves messages is empty, there will be a delay each time the JMS trigger requests a message from the destination. This can impact overall performance.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval

Specifies the length of time (in milliseconds) with which the secondary server threads for a concurrent JMS trigger poll the JMS provider for more messages. The secondary server threads are the additional threads that can be used to retrieve messages for a concurrent JMS trigger. The number of secondary server threads for a concurrent JMS trigger corresponds to the value **Max execution threads** property value minus 1. A short polling interval increases the amount

of CPU used by Integration Server and increases the load on the JMS provider. A long polling interval can impact the time required for Integration Server to shut down as Integration Server must wait for the polling interval to elapse for each secondary thread used by concurrent JMS triggers. The default value is 1 millisecond.

The value of `watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval` must be less than or equal to the value of `watt.server.jms.trigger.concurrent.primaryThread.pollingInterval`.

If a value of 0 is specified, Integration Server uses the `javax.jms.MessageConsumer.receiveNoWait()` API to retrieve messages from the JMS provider. The `javax.jms.MessageConsumer.receiveNoWait()` API may or may not be supported by all JMS providers. If the JMS provider from which a JMS trigger receives messages does not support this API then you may see unexpected behavior. If this happens, modify the value of `watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval` to be a positive integer.

Note:

The default value of 1 millisecond is not optimal when connecting to Universal Messaging. To improve performance, when Universal Messaging is the JMS provider, Integration Server uses a minimum secondary polling interval. If the JMS trigger that receives messages from Universal Messaging uses a join, Integration Server uses a minimum secondary polling interval of 10 milliseconds. If the JMS trigger does not use a join, Integration Server uses a minimum secondary polling interval of 3000 milliseconds. If the configured value is greater than the minimum, Integration Server uses the configured value instead.

Note:

A longer polling interval affects the performance of JMS triggers with joins because the JMS trigger must retrieve messages from multiple destinations. If one of the destinations from which a JMS trigger retrieves messages is empty, there will be a delay each time the JMS trigger requests a message from the destination. This can impact overall performance.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jms.trigger.extendedDelay.delayIncrementInterval

Specifies the time interval, measured in milliseconds, at which Integration Server introduces the polling delay for an inactive concurrent JMS trigger. If the JMS trigger remains inactive, the same time interval determines when Integration Server increases the extended delay. The default value is 0 which indicates that Integration Server does not use an extended polling delay for inactive concurrent JMS triggers. For more information about delaying polling requests, see [“Delaying Polling Requests for Concurrent JMS Triggers” on page 869](#).

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.trigger.extendedDelay.delays

Specifies the length of the delay that an inactive concurrent JMS trigger waits before polling the JMS provider for more messages. To increase the delay while the JMS trigger remains inactive, specify a comma-separated list of increasing integers. The `watt.server.jms.trigger.extendedDelay.delays` parameter is measured in milliseconds and has a default value of: 0, 1000, 2000, 3000.

For more information about delaying polling requests, see [“Delaying Polling Requests for Concurrent JMS Triggers” on page 869](#).

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.trigger.groupTag

Specifies the group tag used in the names of JMS triggers that belong to a trigger group. Integration Server treats JMS triggers with the specified group tag in the name as members of a trigger group. All triggers in a trigger group use the following format for the name: `folder.subfolder:originalJmsTriggerName_groupTag_Id`

The default is WMTG. If you set the parameter to null(blank), Integration Server does not use trigger groups and the methods in the `com.wm.app.b2b.server.jms.consumer.JMSTriggerGroupFacade` class cannot be used to create triggers.

Note:

If you change the value of the `watt.server.jms.trigger.groupTag` parameter and a trigger group already exists on Integration Server, the triggers that were created with the previous group tag will no longer be treated as members of a trigger group.

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.trigger.maxDeliveryCount

Specifies the maximum number of times the JMS provider can deliver a message to Integration Server. You must specify a value greater than or equal to 1. Integration Server ignores any value less than 1. The default is 100.

watt.server.jms.trigger.maxPrefetchSize

Specifies the maximum number of messages that the webMethods Broker API for JMS will retrieve and cache per request for a JMS trigger. Using pre-fetch cache can speed up the retrieval of messages from the webMethods Broker. Because messages will be placed in Integration Server memory, you may want to decrease this setting if you have JMS triggers receiving very large messages. Otherwise, memory issues can occur. This property only applies to JMS triggers receiving messages from the webMethods Broker. The default is 10 messages.

Integration Server checks this value when a JMS trigger starts. To apply a new value to all JMS triggers, use Integration Server Administrator to disable and then enable JMS triggers. For information about using Integration Server Administrator to disable and enable an individual trigger, see [“Managing JMS Triggers” on page 859](#).

This property only takes effect when the trigger's prefetch property is set to -1.

When working in a cluster of Integration Servers, the behavior controlled by this property might appear at first to be misleading. For example, suppose that you have a cluster of two Integration Servers. Each Integration Server contains the same JMS trigger. Twenty messages are sent to a destination from which JMS trigger receives messages. It might be expected the JMS trigger on Integration Server 1 will receive the first message, the JMS trigger on Integration Server 1 will receive the second message, and so forth. However, what may happen is that the JMS trigger on Integration Server 1 will receive the first 10 messages and the JMS trigger on Integration Server 2 will receive the second 10 messages.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.jms.trigger.monitoringInterval

Specifies the interval, measured in seconds, at which Integration Server executes resource monitoring services for JMS triggers. A resource monitoring service is a service that you create to check the availability of resources used by a JMS trigger service. After Integration Server suspends a JMS trigger because all retry attempts have failed, it schedules a system task to execute the resource monitoring service assigned to the JMS trigger. The default is 60 seconds. Changes to this property take effect the next time Integration Server schedules a system task to execute a resource monitoring service for a JMS trigger.

For more information about building resource monitoring services for JMS triggers, see *Using webMethods Integration Server to Build a Client for JMS*.

watt.server.jms.trigger.pollingSession.timeout

Specifies the frequency (measured in minutes) with which Integration Server refreshes an inactive connection for a JMS trigger. A JMS trigger session is considered to be inactive when the JMS trigger does not use the session to retrieve messages. The default is 30 minutes.

watt.server.jms.trigger.pooledConsumer.timeout

Specifies the length of time, in milliseconds, that an inactive pooled consumer remains in the consumer pool before it is removed. Integration Server uses a pool of consumers to retrieve and process messages for concurrent JMS triggers. When the timeout value elapses, Integration Server deletes the inactive pooled consumer. For the Integration Server to retrieve more messages for the concurrent JMS trigger, Integration Server must create a new consumer, which entails creating a `javax.jms.MessageConsumer` and `javax.jms.Session`. For some JMS providers, this can be an expensive operation. Setting a high value for `watt.server.jms.trigger.pooledConsumer.timeout` server property results in the JMS session and `messageConsumer` remaining open for a long period of time. If the load on Integration Server is high, a long timeout value reduces the frequency with which the pooled consumers must be created. However, when the timeout value is high and the load on Integration Server is low there may be unused pooled consumers. The default is 20000 milliseconds.

watt.server.jms.trigger.raiseEventOnException

Indicates whether Integration Server generates a JMS retrieval failure event when a JMS trigger experiences a fatal exception, such as a non-transient error or a message that cannot be reprocessed, during message processing. Specify `true` if you want Integration Server to generate a JMS retrieval failure event. The default is `true`.

watt.server.jms.trigger.raiseEventOnRetryFailure

Indicates that Integration Server generates a JMS retrieval failure event when a JMS trigger experiences a retry failure. A retry failure can occur when the JMS provider has reached the max delivery count for a message or when an `ISRuntimeException` is thrown and the JMS trigger is not configured to recover the message (for example, if retry failure handling for a non-transacted JMS trigger is set to "Throw exception" instead of "Suspend and Retry Later".) When set to `true`, Integration Server generates a JMS retrieval failure event for a JMS trigger when retry failure occurs. The default is `true`.

watt.server.jms.trigger.retryOnConsumerError

Determines whether or not Integration Server attempts to re-enable JMS triggers automatically when an error unrelated to the connection to the JMS provider causes the JMS trigger to be disabled. Set `watt.server.jms.trigger.retryOnConsumerError` to true to instruct Integration Server to re-enable JMS triggers automatically. Set the property to false to instruct Integration Server to leave JMS triggers disabled when the message consumer encounters an unexpected error unrelated to the connection to the JMS provider. The default is true.

watt.server.jms.trigger.reuseJmsTxSession

Specifies whether or not sessions can be reused across multiple transactions for a transacted JMS trigger. When set to true, a transacted JMS trigger will reuse the same JMS session for receiving messages from the JMS provider. When set to false, Integration Server creates and closes a session for each message a transacted JMS trigger receives and processes. The default value is true.

Some JMS providers might not support the use of a single session across multiple transactions. When working with these JMS providers, set `watt.server.jms.trigger.reuseJmsTxSession` to false.

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.trigger.reuseSession

Indicates whether instances of a JMS trigger use the same session on Integration Server. When this property is set to true, the JMS trigger uses a shared session. Each of the trigger services executed by the JMS trigger will use the same session ID. When this property is set to false, Integration Server uses a new session for each instance of a JMS trigger. Reusing sessions for a JMS trigger might improve performance. However, this property does not work with all adapters. If you are working with an adapter that requires the session ID to be unique, set this property to false. The default is false.

Note:

If you use the Adapter for SAP set `watt.server.jms.trigger.reuseSession` to false if a concurrent JMS trigger has a trigger service that executes multi-step SAP transactions that make calls to the `pub.sap.client.lockSession` and `pub.sap.client.releaseSession` services. These services require dedicated sessions.

watt.server.jms.trigger.serial.primaryThread.pollingInterval

Specifies the length of time (in milliseconds) for which the server thread for a serial JMS trigger polls the JMS provider for messages. A short polling interval increases the amount of CPU used by Integration Server and increases the load on the JMS provider. However, shorter polling intervals can improve performance under heavy load and in request-reply scenarios. The default is 500 milliseconds.

If a value of 0 is specified, Integration Server uses the `javax.jms.MessageConsumer.receiveNoWait()` API to retrieve messages from the JMS provider. The `javax.jms.MessageConsumer.receiveNoWait()` API may or may not be supported by all JMS providers. If the JMS provider from which a JMS trigger receives messages does not support this API then the JMS trigger will be disabled. If this happens, modify the value of `watt.server.jms.trigger.serial.primaryThread.pollingInterval` to be a positive integer and then enable the JMS trigger.

Note:

The default value of 500 milliseconds is not optimal when connecting to Universal Messaging. To improve performance when Universal Messaging is the JMS provider, Integration Server uses a minimum primary polling interval of 3000 milliseconds. If the trigger uses a join, Integration Server uses a secondary polling interval which is set to 10 milliseconds. Note that the secondary polling interval is not configurable for a serial JMS trigger

Note:

A longer polling interval affects the performance of JMS triggers with joins because the JMS trigger must retrieve messages from multiple destinations. If one of the destinations from which a JMS trigger retrieves messages is empty, there will be a delay each time the JMS trigger requests a message from the destination. This can impact overall performance.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jms.trigger.startupFailure.restartTaskRetryCount

Specifies the maximum number of retry attempts the trigger restart task makes to start the JMS triggers that fail to start when the JMS connection alias starts up. Integration Server makes the initial start attempt for a JMS trigger when starting the JMS connection alias used by the trigger. If any JMS triggers fail to start when the alias starts, Integration Server uses a trigger restart task to retry starting the failed JMS triggers. The restart task, which uses a separate thread from the thread used to start the alias, runs at an interval specified by the `watt.server.jms.trigger.startupFailure.restartTaskRetryInterval` parameter. To use a restart task, you must specify a positive integer greater than or equal to 1. The default is 6 retry attempts. Set `watt.server.jms.trigger.startupFailure.restartTaskRestryCount` to 0 if you do not want Integration Server to attempt to restart JMS triggers that fail to start when a JMS connection alias starts.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jms.trigger.startupFailure.restartTaskRetryInterval

Specifies the number of seconds that the trigger restart task waits between attempts to restart JMS triggers that failed to start when the JMS connection alias started. You must specify a positive integer. The default is 30 seconds.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.jms.trigger.startupFailure.retryCount

Determines the maximum number of retry attempts that Integration Server makes to start a JMS trigger after the trigger fails to start. After the initial trigger startup failure, Integration Server waits an interval of 1000 milliseconds and then retries starting the trigger. If startup of the trigger fails, Integration Server waits 1000 milliseconds and then makes another attempt to start the trigger. Integration Server continues in this way until the JMS trigger starts successfully, the JMS connection alias stops running, or the JMS trigger becomes disabled. When starting the trigger fails after Integration Server makes the maximum number of retry attempts, Integration

Server can take no further action to start the trigger. Integration Server logs an exception which will be visible in the JMS Trigger Management page of Integration Server Administrator. The JMS trigger remains inactive until the problem is resolved and the trigger is manually restarted (i.e. enabled).

The `watt.server.jms.trigger.startupFailure.retryCount` parameter must be set to an integer greater than or equal to 0. When the parameter is set to 0, the default, Integration Server does not make any retry attempts. Excessive retry attempts may cause the JMS connection alias startup time to be delayed.

Important:

If you change the setting of this parameter, you must restart for the changes to take effect.

watt.server.jms.trigger.stopRequestTimeout

Specifies the maximum amount of time, measured in seconds that Integration Server waits after a JMS trigger is disabled before forcing the JMS trigger to stop processing messages. The minimum value is 0. The default is 3 seconds.

Note:

If you set `watt.server.jms.trigger.stopRequestTimeout` to 0, the JMS trigger immediately stops processing messages after a JMS trigger is disabled. In this case, Integration Server stops the trigger immediately and does not wait for any processing to complete. This can result in a `javax.jms.IllegalStateException`.

You can disable a JMS trigger by invoking the `pub.trigger:disableJMSTrigger` service or by using or via the **Settings > Messaging > JMS Trigger Management** screens in Integration Server Administrator. For information about disabling JMS triggers, see [“About JMS Trigger Status and State” on page 861](#).

When you save a JMS trigger in Designer, Integration Server stops and then restarts the trigger. In this situation, Integration Server waits 3 seconds before forcing the JMS trigger to stop processing messages. This value cannot be configured.

Note:

when a JMS trigger is disabled while it is processing messages,

Important:

You must restart Integration Server for the new value to take effect.

watt.server.jms.trigger.wmjms.clientIDSharing

Indicates that Integration Server attempts to receive messages from durable subscribers in a load-balanced fashion. When set to true, Integration Server does the following:

- If the durable subscriber specified by the JMS trigger does not exist, Integration Server creates the durable subscriber on the Broker and enables state sharing. Integration Server also uses the message processing mode to set the Shared State Order mode for the durable subscriber. (A message processing mode of serial maps to a shared state order mode of publisher, a message processing mode of concurrent maps to a shared state order mode of none.)

If the durable subscriber already exists, Integration Server does not make any changes to the durable subscriber when it saves the JMS trigger.

- Indicates to the Broker that client identifiers can be shared by setting the `com.webmethods.jms.clientIDSharing` property within Integration Server to `true`.

This property only applies when the JMS trigger uses a JMS connection alias that connects to the webMethods Broker using the native webMethods API.

The default is `true`.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

Important:

You must restart Integration Server for a new value to take effect.

watt.server.jms.wmjms.lms.readTimeout

Specifies the amount of time (measured in milliseconds) that Integration Server waits for the next portion of an input stream before throwing `WmReadTimeoutException`. The read timeout only applies after Integration Server retrieves the initial piece of the input stream. The default is 30000 milliseconds.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.jndi.searchresult.maxlimit

Specifies the maximum number of records that Integration Server displays when searching for users or groups in the LDAP or central directory and no search criteria are specified. The default is 0. When set to 0, Integration Server displays all available records.

watt.server.jndi.searchresult.pageSize

Specifies the batch size of user groups that Integration Server retrieves from the connected LDAP servers. Use this parameter to limit the page size of LDAP search results. Set this parameter to 0 to retrieve all user groups from the LDAP servers at once. The default value is 1000.

Note:

This parameter was introduced for PIE-64500 in IS_10.5_Core_Fix7.

watt.server.json.allowUnquotedFieldNames

The JSON standard requires that field names be enclosed in double quotes. However, when parsing legacy JavaScript as JSON text it may be helpful to allow unquoted field names as JavaScript does not require field names to be enclosed in double quotes. If this parameter is set to `true`, the `pub.json:jsonStringToDocument` and `pub.json:jsonStreamToDocument` services allow unquoted field names in any supplied JSON text. If this parameter is set to `false`, the services throw a `ServiceException` when encountering unquoted field names. The default is `false`.

The `watt.server.json.allowUnquotedFieldNames` parameter also affects the process of creating document types from a JSON text. If `watt.server.json.allowUnquotedFieldNames` is set to `true` and a JSON text with unquoted fields is used as the source for a document type, the resulting document type contains fields that correspond to the unquoted fields as well as the quoted fields. When `watt.server.json.allowUnquotedFieldNames` is set to `false` and a JSON text with unquoted fields is used as the source for a document type, Designer throws an exception and does not create the document type.

Important:

You must restart Integration Server for a new value to take effect.

watt.server.json.decodeIntegerAsLong

Converts an integer that Integration Server retrieves from JSON content to either a Long or Integer Java wrapper type. If this parameter is set to `true`, Integration Server converts the JSON integer to a Long Java wrapper type. If this parameter is set to `false`, Integration Server converts the JSON integer to a Integer Java wrapper type. The default is `true` (Long).

watt.server.json.decodeNullRootAsEmpty

Converts a null value that Integration Server retrieves from JSON content to either `IData` or empty `IData`. If this parameter is set to `false`, Integration Server converts the null value to `IData`. If this parameter is set to `true`, Integration Server converts the null value to empty `IData` and subsequent encoding of empty `IData` creates a JSON text of `"{}"`. This JSON content is different from the original JSON content (null) as the original null value gets converted to JSON text of `"{}"`. The default is `false`.

watt.server.json.decodeRealAsDouble

Converts a real number that Integration Server retrieves from JSON content to either a Float or Double Java wrapper type. If this parameter is set to `true`, Integration Server converts the real number to a Double Java wrapper type. If this parameter is set to `false`, Integration Server converts the real number to a Float Java wrapper type. The default is `true` (Double).

watt.server.json.decodeRealAsString

Converts a real number that Integration Server retrieves from JSON content to `String`. If this parameter is set to `true`, Integration Server converts the real number to a `String`. If this parameter is set to `false`, Integration Server does not convert the real numbers to `String` instead, real numbers are converted to either Float or Double Java wrapper type depending on the values specified in *decodeRealAsDouble*. The default is `false`.

Note:

Error occurs if both `watt.server.json.decodeRealAsString` and `watt.server.json.decodeRealAsDouble` parameters are set to `true`.

watt.server.json.decode.maxStringLength

Specifies the maximum length of the JSON strings processed by Integration Server. The parameter provides an option for Integration Server to override the limit on maximum string length. The default value is 2147483647 (the maximum possible value for any integer variable in Java). You must restart Integration Server for changes to this parameter to take effect.

Note:

This server configuration parameter is introduced for PIE-82732 in IS_10.5_Core_Fix25.

watt.server.json.optimizeForUniqueKeys

Controls how Integration Server parses JSON documents based on whether the documents have unique keys or duplicate keys. Integration Server can parse JSON documents faster when you set this parameter based on the nature of the documents you expect to receive. If you expect to process JSON documents that have unique keys or only a few duplicate keys, set this property to `true` (the default). If you expect to process JSON documents that have a lot of duplicate keys, set this property to `false`.

Note:

You must restart Integration Server for changes to go into effect.

watt.server.json.prettyPrint

Specifies whether the output of `JSONCoder.encode()` is formatted with carriage returns and indentation to make the output easier to read. When Integration Server responds to a client request with JSON content, the format of that response is controlled by this property. If this parameter is set to `true`, Integration Server encodes JSON with carriage returns and indentation to ease readability. The default is `false`. For more information about `JSONCoder`, see *webMethods Integration Server Java API Reference*.

You can override this setting in individual requests that use the `invoke`, `rest`, or `restv2` directives by including a query parameter of `jsonPrettyPrint=true` or `jsonPrettyPrint=false` in the URL used to invoke a service.

Note:

You cannot override this setting when using other directives, such as `soap` or `ws`.

You can also set `watt.server.json.prettyPrint` to specify whether the `pub.json:documentToJSONString` service uses pretty print formatting in the `jsonString` output parameter. For more information about `pub.json:documentToJSONString`, see the *webMethods Integration Server Built-In Services Reference*.

watt.server.json.quoteFieldNames

Specifies whether or not the `pub.json:documentToJSONString` service encloses all generated JSON field names in double quotes. The JSON standard requires that field names be enclosed in double quotes. However, you may need the service to produce unquoted field names if the generated JSON text will be processed by an older JavaScript interpreter. Set this parameter to `true` instruct the `pub.json:documentToJSONString` service to enclose field names in quotes in the output JSON text. Set this parameter to `false` to instruct the service to omit the double quotes around field names in the generated JSON text. The default is `true`.

Note:

Use caution when setting `watt.server.json.quoteFieldNames` to `false` as this causes the `pub.json:documentToJSONString` service to generate non-standard JSON text. This can cause interoperability issues if the JSON text is sent to other organizations.

Note:

You must restart Integration Server for changes to take effect.

watt.server.key

Specifies the license key for the server. There is no default.

watt.server.ldap.checkLocalUserInLDAP

Specifies whether Integration Server should authenticate a local user through the connected LDAP servers or not. Set this parameter to `true` for Integration Server to authenticate a user through the connected LDAP servers, even when the username exists in the local system. Set this parameter to `false` to skip LDAP-based authentication if the username exists in the local system. The default value of this parameter is `false`.

Note:

This parameter was introduced for PIE-65332 in IS_10.5_Core_Fix7.

watt.server.ldap.cleanContext

Specifies whether Integration Server should close the TCP/IP connection and clean the LDAP context (the *connectionHandle* object) after processing a *pub.client.ldap* service request. When this property is set to `true`, Integration Server closes the connection and cleans the underlying LDAP context; Integration Server does not return the LDAP context to the pipeline. When this property is set to `false` (the default), Integration Server does not close the connection and returns the LDAP context to the pipeline.

For more information about the *pub.client.ldap* services, see *webMethods Integration Server Built-In Services Reference*.

watt.server.ldap.DNescapeChars

Specifies the characters Integration Server can escape (using the `\` character) in distinguished names presented to an LDAP server. By default, this property is set to `null`, meaning no characters are escaped.

Use this parameter if the distinguished names you are using include special characters that Integration Server needs to escape. For example, if your LDAP server maintains userids such as `abc/def`, specify the following:

```
watt.server.ldap.DNescapeChars = /
```

If you need to escape the `\` character, specify it twice:

```
watt.server.ldap.DNescapeChars = \\
```

watt.server.ldap.DNescapePairs

Specifies the characters in LDAP distinguished names that Integration Server should *not* double-escape prior to their presentation to an LDAP server. To escape special characters in a distinguished name, you must precede the character with a backslash (`\`). By default, the following characters are never double-escaped, and should not be specified in `watt.server.ldap.DNescapePairs`:

```
, = + < > # ; / \
```

If the distinguished names in your LDAP server contain special characters other than those specified above, add them to `watt.server.ldap.DNescapePairs` so that Integration Server does not double-escape the characters. For example, do not add an equal sign (`=`) to `watt.server.ldap.DNescapePairs`, since this is never double-escaped by default. However, if the LDAP distinguished name contains an ampersand (`&`) and it should be double-escaped, add an ampersand character to `watt.server.ldap.DNescapePairs` as follows:

```
watt.server.ldap.DNescapePairs=&
```

This property has no default value. If no value is supplied, no special characters beyond the set listed above are protected from double-escape. This property is not dynamic. Changes to its value go into effect after Integration Server is restarted.

watt.server.ldap.DNescapeURL

Indicates whether Integration Server should ignore the first three forward slashes (`/`) in domain names, or to treat them as escape characters. If set to `false` (the default), Integration Server ignores the first three slashes in a domain name. Integration Server treats the slashes as part of the domain name, not as escape characters, regardless of the escape characters defined in the `watt.server.ldap.DNescapeChars` and `watt.server.ldap.DNescapePairs` server configuration parameters.

If set to `true`, Integration Server treats the forward slashes as escape characters and escapes domain names in accordance to the settings of `watt.server.ldap.DNescapeChars` and `watt.server.ldap.DNescapePairs`.

Important:

You must restart Integration Server after you modify the value of this property.

watt.server.ldap.DNstripQuotes

Specifies whether Integration Server should strip leading and trailing quotes from domain names presented to an LDAP server. When set to `true` (the default), Integration Server strips the quotes from domain names that contain special characters.

Note:

If `watt.server.ldap.DNstripQuotes` is set to `false`, Integration Server retains the quotes in the domain names that have them. This could cause lookups to break.

Important:

You must restart Integration Server after you modify the value of this property.

watt.server.ldap.doNotBind

Specifies whether the Integration Server authenticates against the LDAP server. If your Integration Server uses a custom authentication module and you do not require users to be authenticated against the LDAP directory, set this property to `true` to prevent unnecessary requests to the LDAP server. The default is `false`.

watt.server.ldap.extendedMessages

Controls whether the Integration Server displays additional information returned from the LDAP server when an authentication error occurs. This information is available only if the LDAP server provides it. Active Directory is an LDAP server that provides this additional information. The default is `false`.

When set to `false`, an error message might look like this:

```
2005-03-08 15:40:33 EST [ISS.0002.0035E] Invalid credentials connecting to
ldap://10.3.33.203:389/dc=KQA,dc=webMethods,dc=com as
CN=bob,OU=ISUsers,DC=KQA,DC=WEBMETHODS,DC=COM
```

When set to `true`, the same error would be displayed like this:

```
2005-03-08 15:40:33 EST [ISS.0002.0035E] Invalid credentials connecting to
ldap://10.3.33.203:389/dc=KQA,dc=webMethods,dc=com as
CN=bob,OU=ISUsers,DC=KQA,DC=WEBMETHODS,DC=COM: [LDAP: error code 49 -
80090308: LdapErr: DSID-0C09030F, comment: AcceptSecurityContext error,
data 52e, vece]
```

For Active Directory users, the data code (`data 52e` above) contains the reason the authentication failed. You can convert the code to decimal and look it up on http://msdn.microsoft.com/library/en-us/debug/base/system_error_codes.asp to determine the problem.

watt.server.ldap.extendedProps

Specifies LDAP environment properties that the Integration Server will pass directly to an LDAP implementation when initializing a JNDI context. It takes this form:

For example, if you are using a specialized JNDI provider other than the default, or if your LDAP directory requires a special JNDI property to be passed into the environment when a context is created, you could set the property `customProperty` to `customValue`:

```
watt.server.ldap.extendedProps=java.naming.customProperty=customValue
```

There is no default.

watt.server.ldap.groupSearchFilter

Specifies the filter that Integration Server and SoftwareAG Platform Manager (SPM) plugin use when searching for user groups in the connected LDAP servers. This parameter helps to narrow down the search results and has no default value.

Note:

This parameter was introduced for PIE-64500 in IS_10.5_Core_Fix7.

watt.server.ldap.groupObjectClass

Specifies the LDAP object class for group filter. The default value is `group`. The other allowed values are `groupofnames` and `groupofuniqueNames`.

Note:

This parameter was introduced for PIE-80101 in IS_10.5_Core_Fix22.

watt.server.ldap.includeOnlyActiveGroups

Specifies whether empty groups can be associated with an ACL. Set to `true` if only LDAP groups with at least one member can be associated with an ACL. to `false` if any LDAP group, even empty ones, can be associated with an ACL. The default is `true`.

Note:

In Integration Server Administrator, you can enter search criteria for locating the groups that you want to assign to the ACL. When `watt.server.ldap.includeOnlyActiveGroups` is set to `false`, the search results return any LDAP element (not just groups) that matches the criteria.

watt.server.ldap.memberInfoInGroups

Controls where Integration Server looks for LDAP group membership information. When set to `true`, the default, the Integration Server looks for group membership information in the group object. When set to `false`, the Integration Server looks for group membership information in the user object.

watt.server.ldap.retryCount

Specifies how many times Integration Server should automatically try to reconnect to an LDAP server after a network outage or LDAP server restart. If set to 0, the default, Integration Server will prompt the LDAP user for credentials rather than retrying the connection. If set to a positive integer, Integration Server will retry the connection the number of times specified. The default is 0.

watt.server.ldap.retryWait

Specifies how many milliseconds Integration Server waits before trying to reconnect to an LDAP server after a network outage or LDAP server restart. When set to 0, if there is a transient failure while communicating with LDAP, Integration Server will not try to reconnect to the LDAP server. If set to a positive integer, Integration Server will retry the connection the number of

times specified in `watt.server.ldap.retryCount` and will wait the amount of time specified in `watt.server.ldap.retryWait` between retry attempts. The default is 0.

watt.server.ldap.userObjectClass

Specifies the LDAP object class for user filter. The default value is `user`. The other allowed values are `person`, `inetOrgPerson`, and `organizationalPerson`.

Note:

This parameter was introduced for PIE-80101 in IS_10.5_Core_Fix22.

watt.server.licenses

Specifies the number of licenses. The default is 1.

watt.server.log.alertMaxEntries

Specifies the number of entries that you can display in the **Logs > logName** page beyond which Integration Server Administrator displays an alert message. If the value entered in the **Number of entries to display** field in the **Logs > logName** page is more than the value specified for `watt.server.log.alertMaxEntries`, Integration Server Administrator displays a message informing that the number of requested entries exceeds the threshold and that displaying more entries might affect Integration Server performance. If no value is specified for `watt.server.log.alertMaxEntries`, Integration Server Administrator does not display any alert message.

watt.server.log.maxEntries

Specifies the default number of log entries to be displayed in the log viewing utility. The default is 35 entries (the most recent entries). For complete information, see [“Changing the Display Permanently for All Logs” on page 226](#).

watt.server.log.orphanLoggers

Specifies the orphan loggers that you want to be adopted by the IS logging hierarchy. Some Integration Server logs become filled with debug messages even though the logging level is set to be higher than debug. This can happen when Integration Server has orphan loggers (loggers that are not part of the IS logging tree hierarchy) and a custom application configures the root logger of `log4j`. If there is not a custom configuration of `log4j`, the orphan loggers will inherit the logging levels specified in the IS logging tree hierarchy. However, if a custom application reconfigures the root logger in the `log4j` configuration file, the orphaned loggers inherit the logging level from the root logger. Consequently, if the `log4j.rootLogger` is set to `DEBUG`, the orphaned loggers will log at the Debug level. Use this parameter to prevent orphan loggers from logging at the modified `log4j.rootLogger` level. Use a comma to separate multiple orphan loggers. For example:

```
watt.server.log.orphanLoggers=WEBMETHODS.DEFAULT,  
log4j.logger.com.softwareag.wsstack,log4j.logger.com.softwareag.security,  
Debug.com.webmethods.lwq,com.webmethods.portal.jms.db.DbJMSClient
```

This is the default value.

watt.server.log.queued

Specifies whether the server is to queue log entries written by its facilities in memory, then use a background thread to write them to the server log. The default is `true` (queue log entries). For complete information, see the *webMethods Audit Logging Guide*.

watt.server.log.refreshInterval

Specifies the length of time (in seconds) that Integration Server Administrator refreshes the displayed server log. The default is 90 seconds. For complete information, see [“Changing the Display Permanently for All Logs” on page 226](#).

watt.server.logEncoding

Specifies the encoding that Integration Server uses for reading and writing text to the log files or to the console. The default is UTF-8.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.logRotateInterval

Specifies the length of the log recycle interval (in milliseconds) for log entries. Starting with Integration Server 9.7, this server configuration parameter has been renamed `watt.server.statsLogRotateInterval`.

The `watt.server.logRotateInterval` parameter was removed from Integration Server after 8.2 SP2. When it was reintroduced for the following fixes, the scope of the parameter changed so that it affected *only* the `stats.log`:

- IS_9.0_SP1_Core_Fix6
- IS_9.5_SP1_Core_Fix3
- IS_9.6_Core_Fix2

If you migrate from Integration Server 9.6 or earlier to Integration Server 9.7 or higher, Integration Server:

- Changes the name of the server configuration parameter in `server.cnf` from `watt.server.logRotateInterval` to `watt.server.statsLogRotateInterval`
- Converts the value of the server configuration parameter from milliseconds to minutes
- Uses `watt.server.statsLogRotateInterval` to set the log recycle interval for the `stats.log` file

watt.server.loginFailureLimit

Specifies the number of times a user can supply the wrong login credentials before Integration Server alerts the administrator and resets the count of login failures to 0. For example, if `watt.server.loginFailureLimit` is set to 5, when a user supplies the wrong login credentials for the sixth time, Integration Server sends an email alert to the administrator and resets the count of login failures to 0. The default value is 5.

watt.server.login.userFtpDir

Specifies whether the user who has connected to Integration Server through an FTP listener is to be placed in the default FTP root directory or the client user directory. When this property is set to `false` (or not specified), the user is logged into the FTP root directory. The user must then issue the `cd` command to access the client user directory. When this property is set to `true`, the user is logged into the client user directory. The default value is `false`.

The FTP root directory can be the default directory named `userFtpRoot` in your Integration Server home directory or a directory defined by the user using the `watt.server.userFtpRootDir` property.

Important:

You must restart Integration Server after you modify the value of this property.

watt.server.logs.dateStampFmt

Specifies the format of the date and timestamp to be used in audit log files (FailedAudit_*, WMERROR*, WMSESSION*, WMTXIN*, WMTXOUT*). The format you specify must adhere to the `java.text.SimpleDateFormat` class. If the `watt.server.logs.dateStampFmt` property is not set, Integration Server uses the default format, which is `yyyy-MM-ddThh:mm:ss.SSSZ` (for example, 2010-04-19T19:07:21.505Z).

watt.server.logs.dateStampTimeZone

Specifies the time zone to be used in audit log files (FailedAudit_*, WMERROR*, WMSESSION*, WMTXIN*, WMTXOUT*). The format you specify must adhere to the `java.util.TimeZone` class (for example, `watt.server.logs.dateStampTimeZone=EST`). If this property is not set, Integration Server uses local time zone hosting. For this property to take effect, the `watt.server.logs.dateStampFmt` must also be specified.

watt.server.math.floatOperation.mode

Specifies whether the `pub.math:*Floats` services return the exact result of an operation involving two floating point numbers, the result as calculated by the JVM, or the result based on a fixed number of decimal places.

Set this parameter to one of the following values:

Specify	<code>pub.math:*Floats</code> services to
dynamic	Return the precise result of the math operation. For example, when using <code>pub.math:addFloats</code> to add 62.98 and 23, the service returns a sum of 85.98.
default	Return the result of the math operation as calculated by the JVM. For example, when using <code>pub.math:addFloats</code> to add 62.98 and 23, the service returns a sum of 85.97999999999999. This is the default behavior.
fixed	Return the result rounded to a pre-determined number of decimal places. For the <code>pub.math:addFloats</code> and <code>pub.math:subtractFloats</code> services, Integration Server rounds the result to 3 decimal places. For the <code>pub.math:multiplyFloats</code> service, Integration Server rounds the product to 16 decimal places. For the <code>pub.math:divideFloats</code> service, Integration Server rounds the result to 17 decimal places.

Note:

If you specify a value for the *precision* input parameter in `pub.math:*Floats` services, the *precision* value is given precedence over the value of the property `'watt.server.math.floatOperation.mode`. For more information about the `pub.math:*Floats` services, see *webMethods Integration Server Built-In Services Reference*.

Important:

You must restart Integration Server for a new value to take effect.

watt.server.mediator.directives

Specifies a comma-separated list of directives used to detect API Gateway requests. When Integration Server receives a request that uses the directives specified by the `watt.server.mediator.directives` server configuration parameter, Integration Server checks to see if the web service invoked by the request is a API Gateway service. If the web service is a API Gateway service, Integration Server assigns the default user to the request. The default API Gateway directives are `"ws,mediator,gateway"`.

watt.server.messaging.csq.maxRedeliveryCount

Specifies the maximum redelivery attempts Integration Server makes when publishing a message from the client side queue (CSQ) to Universal Messaging. When Integration Server publishes messages and the Universal Messaging is not available, Integration Server writes guaranteed messages to the CSQ if one is configured for use with the Universal Messaging connection alias. When Universal Messaging becomes available, Integration Server drains the CSQ by publishing message from the CSQ to Universal Messaging. If the initial attempt to publish the message to Universal Messaging from the CSQ fails, Integration Server makes subsequent attempts until the message is published successfully or Integration Server makes the maximum redelivery attempts specified by this parameter. Each attempt to publish to Universal Messaging from the CSQ is considered a redelivery attempt. If Integration Server makes the maximum redelivery attempts and all attempts have failed, Integration Server discards the document. The `watt.server.messaging.csq.maxDeliveryCount` parameter must be set to a positive integer. The default value is 5.

Note:

This parameter applies to webMethods messaging where Universal Messaging is the messaging provider. The comparable parameter for webMethods messaging where Broker is the messaging provider is `watt.server.publish.maxCSQRedeliveryCount`. The comparable parameter for JMS messaging, regardless of provider, is `watt.server.jms.csq.maxRedeliveryCount`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.messaging.csq.publishDelayWhileDraining

Specifies the number of seconds that Integration Server waits before publishing a message to the client side queue (CSQ) while the CSQ drains. To use the delay when the CSQ contains one or more messages, specify an integer greater than 0 but less than or equal to 60.

Specify 0 to instruct Integration Server to vary the length of the delay depending on the number of messages in the CSQ. When `watt.server.messaging.csq.publishDelayWhileDraining` is set to 0, Integration Server waits for:

- 6 seconds when the number of messages in the CSQ is greater than 0 but less than or equal to 1000.
- 8 seconds when the number of messages in the CSQ is greater 1000 but less than or equal to 5000.

- 10 seconds when the number of messages in the CSQ is greater than 5000

The default value for `watt.server.messaging.csq.publishDelayWhileDraining` is 0.

The `watt.server.messaging.csq.publishDelayWhileDraining` applies to `webMethods` messaging for which the messaging provider can be `Broker` or `Universal Messaging`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.messaging.debugTrace

Enables an extra level of verbose logging for `webMethods` messaging triggers that receive messages from `Universal Messaging` or through `Digital Event Services`. You can configure the additional logging globally or at the individual `webMethods` messaging trigger level.

- To enable `debugTrace` logging for all `webMethods` messaging triggers, set the `watt.server.messaging.debugTrace` property to `true`. The default value is `false`.

For changes to this parameter to take effect, you need to disable and then enable the messaging connection aliases used by the triggers.

- To enable `debugTrace` logging for a specific `webMethods` messaging trigger, append the fully qualified `webMethods` messaging trigger name to the `watt.server.messaging.debugTrace` property and set that property to `true`. For example, to enable `debugTrace` logging for the `webMethods` messaging trigger named `myFolder:myMessagingTrigger`, enter the following on the `Edit Extended Settings` page:

```
watt.server.messaging.debugTrace.myFolder:myMessagingTrigger=true
```

For this parameter or changes to this parameter to take effect, you need to disable and then enable the `webMethods` messaging trigger specified in the property. For information about using `Integration Server Administrator` to disable and enable an individual trigger, see [“Suspending or Resuming Document Processing for a Specific webMethods Messaging Trigger”](#) on page 845

Note:

For the increased logging to appear in the server log, you must set the logging level for server facility `0153 Dispatcher (Universal Messaging)` to `Trace`.

watt.server.metadata.registry.timeout

The number of minutes a connection to a `CentraSite` registry can remain inactive before `Integration Server` closes it. `Integration Server` connects to one or more `CentraSite` registries to publish and retract metadata for your `Integration Server` assets. If no assets are published to or retracted from a `CentraSite` registry for a period of time, `Integration Server` closes the connection to that registry. The default period is 10 minutes.

watt.server.metering.simulator.enabled

Specifies whether the metering simulator is enabled. Set this parameter to `true` to have `Integration Server` gather metering data for top-level services and write metering simulation messages to the server log. Set to `false` to disable the metering simulator. The default is `false`.

For more information about the metering simulator, see [“Simulating Metering in Integration Server”](#) on page 981.

Note:

The `watt.server.metering.simulator.enabled` parameter is added for PIE-67717 (IS_10.5_Core_Fix9).

watt.server.meteringAgent.logInterval

Specifies the interval, in minutes, between executions of the system task Metering Agent Diagnostic Log which performs a "dry run" of sending data to the webMethods Metering Server. Specify a positive integer. The default is 60 minutes.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

The `watt.server.meteringAgent.logInterval` parameter is added for PIE-69024 (IS_10.5_Core_Fix9).

watt.server.mime.decodeHeaders

Determines the global default behavior for how the `pub.mime:createMimeData` service handles header decoding. The property can have one of the following values:

Specify	To indicate that
NONE	Specifies that by default the <code>pub.mime:createMimeData</code> service does not decode the MIME header or body part headers. This is the default.
ONLY_MIME_HEADER	Specifies that by default the <code>pub.mime:createMimeData</code> service decodes the MIME header only.
ONLY_BODY_PART_HEADERS	Specifies that by default the <code>pub.mime:createMimeData</code> service decodes the body part headers only.
BOTH	Specifies that by default the <code>pub.mime:createMimeData</code> service decodes the MIME header and the body part headers.

Note:

The values are not case sensitive.

Note: Integration Server uses the value of the `watt.server.mime.decodeHeaders` to determine whether or not to decode headers in a MIME message only when the `pub.mime:createMimeData` service specifies no value for the `decodeHeaders` input parameter. For information about the `pub.mime:createMimeData` service and the `decodeHeaders` input parameter, see the *webMethods Integration Server Built-In Services Reference*.

watt.server.mqtt.producer.maxInflight

Specifies the maximum number of MQTT messages that can be published at one time using the same MQTT connection alias. The default is -1 which means that Integration Server relies on the Paho client to set the max inflight property. Paho uses a default value of 10 for the max inflight property. Set the `watt.server.mqtt.producer.maxInflight` property to a value greater

than 0 to use the value set by the configuration parameter as the value of the max inflight property.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

The `watt.server.mqtt.producer.maxInflight` parameter is introduced in PIE-60449 (IS_10.5_Core_Fix8)

watt.server.netEncoding

Specifies the encoding the server is to use when reading and writing text to the network. This setting has no effect on text that is explicitly encoded in a particular encoding. The default is UTF-8. The *encoding* parameter in the `pub.client:soapHTTP` service, if set, overrides the `watt.server.netEncoding` setting. For more information about `pub.client:soapHTTP`, see *webMethods Integration Server Built-In Services Reference*.

When the `pub.client:http` service submits a password digest for authentication (that is, the *auth/type* field is set to `Digest`) and the HTTP server response includes the header field "Content-Type" but does not contain the charset parameter, Integration Server uses the value of the `watt.server.netEncoding` server configuration parameter as the default character set.

watt.server.new.http.session.context

Specifies whether Integration Server should create a new session object when executing the `pub.client:http` service. When this property is set to `true`, Integration Server ignores the values from the session object of the last invocation of `pub.client:http` and creates a new session object. When this property is set to `false`, Integration Server uses the values from the existing session object. The default value is `false`.

watt.server.noObjectURL

Specifies the URL to which the server redirects a request after three attempts to log on to the Integration Server Administrator have failed because the server cannot find the document the user is requesting. The default is for the server to display an HTML screen saying "No such object."

watt.server.noAccessURL

Specifies the URL to which the server is to redirect a request after three attempts to log on to the Integration Server Administrator have failed because the user does not have access to the requested document. The default is for the server to display an HTML screen saying "Access denied."

watt.server.ns.backupNodes

Specifies whether services are removed completely when they are deleted. When set to `true`, service `node.ndf` files will be renamed to `node.bak` when they are deleted. The default is `false`.

watt.server.ns.dependencyManager

Specifies whether the dependency checking features are enabled or disabled. When set to `true`, the dependency checking features identify elements that will be affected by moving, renaming, or deleting another element. For optimization in a production environment, you might set this parameter to `false`. The default for this parameter is `true`.

watt.server.ns.decodeJavaService

Specifies whether Integration Server decodes and Designer displays non-ASCII Unicode characters in the body of a Java service. When you save Java services, Integration Server encodes non-ASCII Unicode characters as ASCII Unicode escape sequences. By default, Integration Server does not decode the escape sequences. Therefore, when Integration Server sends Java service code to Designer, escape sequences appear in Designer instead of the original Unicode characters. Set to `true` if you want Integration Server to decode the escape sequences so that Designer displays the original Unicode characters instead of the escape sequences. The default is `false`.

watt.server.ns.hideWmRoot

Indicates whether Integration Server should hide the `WmRoot` package from the Software AG Designer workspace. When set to `true` (the default), Designer does not display the `WmRoot` package in the workspace.

watt.server.ns.lockingMode

Specifies whether file locking is enabled on the Integration Server:

- To enable local locking on the Integration Server, set this value to `full`. This is the default value.
- To disable user locking and show no locks, set this value to `none`.
- To disable user locking but show system locks, set this value to `system`. This value is required to use the Local Service Development feature.

watt.server.ns.logDuplicateDocTypeRegistrationAsError

Indicates whether to suppress or continue logging of error messages related to registration of duplicate universal names for a document type.

When the same WSDL document is used to generate multiple consumer Web service descriptors and multiple WSDL first provider Web service descriptors, Integration Server creates multiple document types with the same explicit universal name. It is when loading a package containing these document types that Integration Server logs error messages specifying duplicate universal name registrations multiple times, which can clutter the log file.

Set the value of the parameter to `false` to suppress the logging of the error messages and `true` to resume the logging. The default value of this parameter is `true`.

Note:

Setting `watt.server.ns.logErrorsOnRegisteringMultipleDocTypesForAUniversalName` to `false` suppresses the error messages about duplicate universal names only. It does not resolve the duplicate names.

watt.server.oauth.approvalpage.footer

Specifies the footer information that Integration Server displays on the OAuth approval page. There is no default.

watt.server.oauth.approvalpage.header

Specifies the heading information that Integration Server displays on the OAuth approval page. The default is `Resource access approval`.

watt.server.oauth.approvalpage.logo

Specifies the URL of the image file that Integration Server displays on the OAuth approval page. Keep the following points in mind when specifying an image file:

- Image files can be of any width, but can be no larger than 92 pixels high.
- You can specify an image of any image file format (for example, .gif, .png, .jpeg, or another image file format) that your browser can display.
- The image file does not have to reside on Integration Server. If the image file is hosted on another server, provide the absolute path.

The default is `/WmPublic/images/fw_logo_sag.gif`, which is 234x92 pixels.

watt.server.oauth.approvalpage.title

Specifies the title that Integration Server displays on the OAuth approval page. The default is `Access Approval`.

watt.server.oauth.authCode.expirySeconds

Specifies the amount of time (in seconds) before the OAuth authorization code expires. A value of -1 indicates that the authorization code never expires. The maximum value is 2147483647. The default is 600.

The value of the `watt.server.oauth.authCode.expirySeconds` parameter is tied to the value of the **Authorization code expiration interval** field on the **Security > OAuth > Edit OAuth Global Settings** screen in Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to specify the expiration time for the OAuth authorization code instead of using the `watt.server.oauth.authCode.expirySeconds` property. For more information about setting the expiration time for the OAuth authorization code, see [“Configuring OAuth” on page 601](#).

watt.server.oauth.authServer.alias

Specifies the alias of the remote server that Integration Server uses as an authorization server. The value should match the **Alias** field on the **Settings > Remote Servers** page of Integration Server Administrator. For example, `local`.

Note:

You must add the remote server on the **Settings > Remote Servers** page of Integration Server Administrator *before* specifying the alias in `watt.server.oauth.authServer.alias`.

The value of the `watt.server.oauth.authServer.alias` parameter is tied to the value of the **Authorization server** field on the **Security > OAuth > Edit OAuth Global Settings** screen in Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to specify the remote server alias to use as an authorization server instead of using the `watt.server.oauth.authServer.alias` property. For more information about specifying the remote server alias, see [“Configuring OAuth” on page 601](#).

watt.server.oauth.custom.responseHeader

When set to true, OAuth and OpenID errors that are returned to client applications and to end users appear in the HTTP response header with a header attribute name of `X-OAuth-Error`. For most OAuth and OpenID errors, Integration Server returns a status code in the range of 400 - 599 and the error message is returned in the body of the HTTP response. Some browsers do not display the response body when the status code is not successful (200 - 299). Problems with your applications that use OAuth or OpenID can be easier to troubleshoot when the error message is in the response header. When the Server logging facility "0038 HTTP Header" is set

to TRACE, the response header is visible in the server log. If `watt.server.oauth.custom.responseHeader` is true, OAuth and OpenID error messages will appear as X-OAuth-Error response header attributes in the server log.

The default value for `watt.server.oauth.custom.responseHeader` is false.

watt.server.oauth.disableClient.disableTokens

Specifies whether a client access token issued to a client account that is disabled can be used to access resources. When set to true, the Integration Server introspection endpoint, the `pub.oauth.introspectToken` service, considers the token to be inactive if the client account to which the token was issued is disabled. The service returns a value of false for the *active* output parameter. The OAuth resource server, whether it's an Integration Server or another vendor, will not allow access to the requested resource. When set to false, Integration Server introspection endpoint does not consider the enabled/disabled state of the client account to which the access token was issued when evaluating an access token. The default is true.

watt.server.oauth.requireHTTPS

Indicates whether Integration Server requires that client applications use HTTPS to access the `pub.oauth*` services. If you set the value of this property to `false`, Integration Server allows client applications to use HTTP to access the `pub.oauth*` services. The default is true.

If you set the value of this property to `true` and the client application accesses any of the `pub.oauth` services over HTTP, Integration Server issues an HTTP 400 error response to the client and writes a service exception to the error log.

Note:

To conform to the OAuth 2.0 Authorization Framework, you must set this property to `true`.

Important:

To simplify development, you can set `watt.server.oauth.requireHTTPS` to `false`, but you should require HTTPS in production in accordance with the OAuth Framework. If you do not require HTTPS, the authorization server transmits access tokens in clear text, making them vulnerable to theft.

The value of the `watt.server.oauth.requireHTTPS` parameter is tied to the value of the **Require HTTPS** field in the **Security > OAuth > Edit OAuth Global Settings** page of the Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to indicate whether Integration Server requires HTTPS for applications to access OAuth services instead of using `watt.server.oauth.requireHTTPS`. For more information about requiring HTTPS for accessing OAuth services, see [“Configuring OAuth” on page 601](#).

For more information about the `pub.oauth` services, see *webMethods Integration Server Built-In Services Reference*.

watt.server.oauth.requirePost

Indicates whether Integration Server requires that client applications use HTTP POST method to access the `pub.oauth*` services. If you set the value of this property to `false`, Integration Server allows client applications to use the HTTP GET method to access the `pub.oauth*` services. The default is true.

If you set the value of this property to `true` and the client application accesses any of the `pub.oauth` services using an HTTP method other than POST, Integration Server issues an HTTP 400 error response to the client and writes a service exception to the error log.

Note:

To conform to the OAuth 2.0 Authorization Framework, you must set this property to true.

Important:

To simplify development, you can set `watt.server.oauth.requirePOST` to `false`, but you should require the HTTP POST method in production in accordance with the OAuth Framework.

watt.server.oauth.token.defaultExpirySeconds

Specifies the amount of time (in seconds) that access tokens are valid before expiring. A value of -1 indicates that authorization tokens never expire. The maximum value is 2147483647. The default is 3600.

The value of the `watt.server.oauth.token.defaultExpirySeconds` parameter is tied to the value of the **Access token expiration interval** field in the **Security > OAuth > Edit OAuth Global Settings** page of the Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to set the expiration time for access tokens instead of using `watt.server.oauth.token.defaultExpirySeconds`. For more information about the **Access token expiration interval** field, see [“Configuring OAuth” on page 601](#).

watt.server.oauth.token.endpoint.auth

Specifies whether the token endpoint accepts an existing session or requires credentials for authentication. Set to "session" to indicate that the token endpoint will accept requests from clients that have an active session on Integration Server. If these clients supply a valid session identified in the Cookie request header, the clients do not have to provide credentials to use the `pub.oauth:getToken`, `pub.oauth:getAccessToken`, and `pub.oauth:refreshAccessToken` services. Set to "credentials" if clients need to provide their credentials in the Authorization request header every time they request a new access token or refresh an existing access token by calling the `pub.oauth:getToken`, `pub.oauth:getAccessToken`, and `pub.oauth:refreshAccessToken` services. The default value is session.

The value of the `watt.server.oauth.token.endpoint.auth` parameter is tied to the value of the **Token endpoint authentication** field in the **Security > OAuth > Edit OAuth Global Settings** page of the Integration Server Administrator. Software AG recommends that you use Integration Server Administrator to set the token endpoint authentication behavior instead of using `watt.server.oauth.token.endpoint.auth`. For more information about the **Token endpoint authentication** field, see [“Configuring OAuth” on page 601](#).

watt.server.oauth.token.endpoint.internal.requireSecret

Specifies whether Integration Server ensures that authorization code provided by a confidential client invoking the OAuth token endpoint service was issued to the confidential client when the OAuth token endpoint service is invoked directly because it is on the same Integration Server acting as the OAuth authorization server. When set to true, Integration Server uses the credentials supplied by the requesting client to ensure that the authorization code provided by the client requesting the OAuth token matches the credentials of the client that requested the authorization code. The `client_id` and `client_secret` must be in the input pipeline passed to the token endpoint service. When set to false, Integration Server does not ensure that the authorization code provided by the client matches the credentials of the client that actually requested the token. The default is false, which is less secure.

Note:

The `watt.server.oauth.token.endpoint.internal.requireSecret` value applies when getting an initial access token and when using a refresh token to get a new access token.

Note:

Public clients doing an invoke of the token endpoint service on the same Integration Server as the redirection endpoint must continue to include their `client_id` in the input pipeline passed to the token endpoint service.

Clients that use HTTP or HTTPS to request an access token from the OAuth token endpoint must continue to send these requests as they have been.

- Confidential clients must use their `client_id` and `client_secret` in the HTTP Authorization header.
- Public clients must include their `client_id` in the body of the request.

Note:

The services that act as the OAuth token endpoint are `pub.oauth:getAccessToken` in Integration Server 10.1 and `pub.oauth:getToken` in later releases.

watt.server.package.maxSizeMB

Specifies the maximum expanded size of a package that can be installed, measured in megabytes. A package whose expanded size is larger than this parameter value cannot be installed. The default value is 10000 (10 gigabytes).

watt.server.package.parallel.threads

Specifies the maximum number of threads that you want Integration Server to use when loading IS packages at startup. The threads are from a thread pool that Integration Server uses only at startup and specifically for package loading. If you want Integration Server to load packages sequentially, set the value to 1. If you want Integration Server to load packages in parallel, set the value to 2 or higher. It is recommended that you do not exceed 10. The default value is 6.

Note:

If you set `watt.server.package.parallel.threads` to 0 or a negative number, Integration Server loads packages sequentially.

watt.server.package.pre82WSD.loadExternalResources

Specifies whether, at package load time, Integration Server loads the external resources for a consumer web service descriptor or a WSDL first provider web descriptor created on a pre-8.2 version of Integration Server with the **Pre-8.2 compatibility mode** property set to true.

When this parameter is set to true, Integration Server resolves all the import and include statements in the source WSDL document and related XML Schema definition files for consumer web service descriptors or WSDL first provider web service descriptors at the package load time.

When this parameter is set to false, Integration Server skips loading the external resources contained in the source WSDL document used to create the web service descriptor. In some environments, setting this parameter to false may speed up the loading of web service descriptors. The default is true.

Note:

This parameter is deprecated because the web services implementation introduced in Integration Server 7.1 is deprecated as of Integration Server 10.4. The web services implementation introduced in Integration Server version 7.1 handles web service descriptors that run in pre-8.2 compatibility mode.

watt.server.partner

Specifies the IP address or domain name of a hub server. This setting allows the partner server to issue a remote invoke request to a remote Integration Server. Integration Server will ignore a port number if you specify one as part of an IP address.

watt.server.password.historyLength

Specifies the maximum number of previously set passwords that Integration Server saves in the history for a user. You cannot choose a password that matches with any of the passwords stored in the memory for a user account. The default value is 0.

The `watt.server.password.historyLength` displays the value specified in the **Number of Old Passwords to Remember** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the history length of passwords using the **Number of Old Passwords to Remember** parameter on the **Security > User Management > Password Restriction > Edit** screen instead of changing the value of the `watt.server.password.historyLength` parameter.

watt.server.password.maxIdenticalCharsInARow

Specifies the maximum number of identical characters in a row a password can contain. When a user creates or changes a password, he or she must create a password that does not exceed the limit of continuous identical characters as specified by this parameter. If the number of identical characters in a row exceeds for the new password, then based on the Password Enforcement Mode Integration Server sends a warning or error message to users based on the value specified for `watt.server.password.mode` server configuration parameter. The default is 3.

The `watt.server.password.maxIdenticalCharsInARow` identifies the value specified in the **Maximum Number of Identical Characters in a Row** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the maximum continuous identical characters in passwords using the **Maximum Number of Identical Characters in a Row** parameter on the **Security > User Management > Password Restrictions > Edit** screen instead of changing the value of the `watt.server.password.maxIdenticalCharsInARow` parameter.

watt.server.password.maxLength

Specifies the maximum number of characters (alphabetic characters, digits, and special characters combined) the password must contain. When a user changes a password, he or she must create a password for which the maximum number of characters cannot exceed beyond the value specified by this parameter. If the value exceeds, Integration Server sends a warning or error message to users based on the value specified for `watt.server.password.mode` server configuration parameter. The default is 64 characters.

The `watt.server.password.maxLength` identifies the value specified in the **Maximum Password Length** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the maximum number of characters in passwords using the **Maximum Password Length** parameter on the **Security > User Management > Password Restrictions > Edit** screen instead of changing the value of the `watt.server.password.maxLength` parameter.

watt.server.password.minDigits

Specifies the minimum number of digits a password must contain for non-administrator users. When a non-administrator user changes a password, he or she must create a password containing at least the number of digits specified by this parameter. If the minimum number of digits is not met, Integration Server sends a warning or error message to users based on the value specified for watt.server.password.mode server configuration parameter. The default is 1.

The watt.server.password.minDigits identifies the value specified in the **Minimum Number of Digits** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the minimum number of digits a password should contain using the **Minimum Number of Digits** parameter on the **Security > User Management > Password Restrictions > Edit** screen instead of changing the value of the watt.server.password.minDigits parameter.

watt.server.password.minLength

Specifies the minimum number of characters a password must contain for non-administrator users. The character length includes upper and lower case alphabetic characters, digits (0-9), and special characters combined. When a non-administrator user changes a password, he or she must create a password containing at least the number of characters specified by this parameter. If the minimum length is not met, Integration Server sends a warning or error message to users based on the value specified for watt.server.password.mode server configuration parameter. The default is 8.

The watt.server.password.minLength identifies the value specified in the **Minimum Password Length** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the minimum number of characters a password must contain using the **Minimum Password Length** parameter on the **Security > User Management > Password Restrictions > Edit** screen instead of changing the value of the watt.server.password.minLength parameter.

watt.server.password.minLowerChars

Specifies the minimum number of lowercase alphabetic characters a password must contain for non-administrator users. When a non-administrator user changes a password, he or she must create a password containing at least the number of lowercase characters specified by this parameter. If the minimum is not met, Integration Server sends a warning or error message to users based on the value specified for watt.server.password.mode server configuration parameter. The default is 2.

The watt.server.password.minLowerChars identifies the value specified in the **Minimum Number of Lowercase Characters** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the minimum number of lowercase alphabetic characters a password must contain using the **Minimum Number of Lowercase Characters** parameter on the **Security > User Management > Password Restrictions > Edit** screen instead of changing the value of the watt.server.password.minLowerChars parameter.

watt.server.password.minSpecialChars

Specifies the minimum number of special characters, such as asterisk (*), period(.), and question mark(?), a password must contain for non-administrator users. When a non-administrator user changes a password, he or she must create a password containing at least the number of special characters specified by this parameter. If the minimum is not met, Integration Server sends a warning message to the Administrator user. The default is 2.

Note:

A password cannot begin with an asterisk (*).

The `watt.server.password.minSpecialChars` identifies the value specified in the **Minimum Number of Special Characters** field on the **Security > User Management > Password Restrictions > Edit** screen in Integration Server Administrator. Software AG recommends that you specify the minimum number of special characters such as asterisk (*), period(.), and question mark(?), a password must contain using the **Minimum Number of Special Characters** parameter on the **Security > User Management > Password Restrictions > Edit** screen instead of changing the value of the `watt.server.password.minSpecialChars` parameter.

watt.server.password.minUpperChars

Specifies the minimum number of uppercase alphabetic characters a password must contain for non-administrator users. When a non-administrator user changes a password, he or she must create a password containing at least the number of uppercase characters specified by this parameter. If the minimum is not met, Integration Server sends a warning or error message to users based on the value specified for `watt.server.password.mode` server configuration parameter.

watt.server.password.mode

Specifies whether or not to enforce the password restrictions when passwords are set by administrator or non-administrator users. When set to `strict`, the password restrictions are enforced on all users and when set to `lax`, the password restrictions are enforced on non-administrator users. The default is `lax`.

watt.server.pipeline.processor

Specifies whether to globally enable or disable the Pipeline Debug feature. When set to `true`, Integration Server checks the Pipeline Debug values for the service at run time. You can view and set the Pipeline Debug values in the Properties view in Designer. When set to `false`, Integration Server ignores the Pipeline Debug options set for the service in Designer. The default is `true`.

Enable this property in development environments where testing and debugging is performed. In a production environment, however, disabling this property is recommended.

Important:

You must restart Integration Server for the new value to take effect.

watt.server.port

Specifies the port number of the Integration Server's primary port. The default is 5555.

watt.server.portAccess.axis2

Specifies whether Integration Server verifies that an Axis2-based web service can be accessed through a port. Set the parameter to `true` if you want Integration Server to perform the access check on Axis2-based web services. Set this parameter to `false` to disable the access check for Axis2-based web services. The default is `true`.

watt.server.portQueue

Specifies the size of the port queue for HTTP and HTTPS ports. The port queue is the number of outstanding inbound connections that are queued in the TCP/IP stack. The default is 65534. If your server runs on AS/400, set this number to 511.

watt.server.ports.ipaccess.ignoreXForwardedForHeader

Specifies whether Integration Server uses or ignores the IP address in the X-Forwarded-For (XFF) header of a request when determining whether to allow or deny access to the port.

When set to true, Integration Server does not use the IP address in the XFF header to determine if the request is from a host allowed to access the port. Instead, Integration Server uses the IP address of the proxy or load balancer.

- If a request is sent through a proxy or a load balancer from a host whose IP address is on the Deny List for the port configured to Allow by Default and the IP address of the proxy/load balancer is not blocked, the host will not be blocked either. Instead, the host will have access to the port.
- If a request is sent through a proxy or a load balancer from a host whose IP address is on the Allow List for a port configured to Deny by Default and the IP address of the proxy/load balancer is blocked, the host will not have access to the port.

When set to false, Integration Server uses the IP address in the XFF header to determine if a request originates from a host allowed to access the port.

- If a request is sent through a proxy or a load balancer from a host whose IP address is on the Deny List for a port configured to Allow by Default, the host will not have access to the port.
- If a request is sent through a proxy or a load balancer from a host whose IP address is on the Allow List for a port configured to Deny by Default, the host will have access to the port.

The default value is true.

watt.server.portStateless

Specifies a comma-separated list of the port numbers for the ports on Integration Server that are stateless. Integration Server does not provide or maintain any sessions or session IDs for requests received on the port.

You can configure any HTTP or HTTPS port on Integration Server to be stateless with the exception of the primary port and the diagnostic port. Additionally, do not configure any ports used for WebSockets or for administering Integration Server to be stateless. If you specify the primary port, diagnostic port, a port that is not an HTTP or HTTPS port, or a port that does not exist in the value for `watt.server.portStateless`, Integration Server logs a warning message stating that the port cannot be made stateless and that it will be skipped.

There is no default value for this parameter. That is, by default, none of the ports on Integration Server are stateless.

watt.server.publish.drainCSQInOrder

If client-side queuing is enabled, this parameter specifies whether Integration Server should empty the outbound document store in publication order or in parallel with newly published documents. When this parameter is set to `true`, Integration Server sends all newly published documents (guaranteed and volatile) to the outbound document store until the outbound store has been emptied. This allows the Integration Server to maintain publication order. When this parameter is set to `false`, the outbound store is emptied while new documents are being published to the Broker. The default is `true`.

Note:

This server configuration parameter applies to documents published to Broker only.

Note:

This parameter is deprecated because webMethods Broker is deprecated.

watt.server.publish.local.rejectOOS

Specifies whether Integration Server should reject locally published documents when the queue for the subscribing trigger is at maximum capacity.

When this parameter is set to `true`, before placing a locally published document into a subscribing trigger's queue, Integration Server first checks the trigger's queue size. If the queue already contains the maximum number of documents allowed by the trigger's **Capacity** property, Integration Server rejects the locally published document for that trigger queue only.

When this parameter is set to `false`, Integration Server continues to place locally published documents into a subscribing trigger's queue even when the queue is at capacity. This is the default.

Note:

Multiple triggers can subscribe to the same document. Integration Server places the document in any subscribing trigger queue that is not at capacity.

Note:

This parameter applies only to documents published locally using the `pub.publish:publish` or `pub.publish.publishAndWait` services.

watt.server.publish.maxCSQRedeliveryCount

Specifies the maximum redelivery attempts Integration Server makes when publishing a message from the client side queue (CSQ) to the Broker. When Integration Server publishes messages and the Broker is not available, Integration Server writes guaranteed messages to the CSQ if one is configured for use with the Broker (`watt.server.publish.useCSQ = always`). When the Broker becomes available, Integration Server drains the CSQ by publishing message from the CSQ to the Broker. If the initial attempt to publish the message to Broker from the CSQ fails, Integration Server makes subsequent attempts until the message is published successfully or Integration Server makes the maximum redelivery attempts specified in `watt.server.publish.maxCSQRedeliveryCount`. Each attempt to publish to Broker from the CSQ is considered a redelivery attempt. The `watt.server.publish.maxCSQRedeliveryCount` parameter must be set to a positive integer. The default value is 5.

Note:

This parameter applies to webMethods messaging where Broker is the messaging provider. The comparable parameter for webMethods messaging where Universal Messaging is the messaging provider is `watt.server.messaging.csq.maxRedeliveryCount`. The comparable parameter for JMS messaging, regardless of provider, is `watt.server.jms.csq.maxRedeliveryCount`.

Note:

This parameter is deprecated because Broker is deprecated.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.publish.useCSQ

Specifies whether Integration Server uses outbound client-side queuing if documents are published when the Broker is unavailable. Set this parameter to `always` to send published documents to the outbound document store when the Broker is not available. Set this parameter to `never` to instruct the publishing service to throw a `ServiceException` when the Broker is not available. The default is `always`.

Note:

This parameter is deprecated because `webMethods Broker` is deprecated.

watt.server.publish.usePipelineBrokerEvent

Specifies whether Integration Server should bypass encoding that is normally performed when documents are published to the Broker. If this property is set to `true`, Integration Server checks the pipeline for an object called `$brokerEvent`. If the object is found and is of type `BrokerEvent`, Integration Server sends its value to the Broker and no additional encoding is performed. Set this parameter to `true` if Integration Server is sending native Broker events. The default is `false`. For more information about publishing native Broker events, see the *Publish-Subscribe Developer's Guide*.

Note:

This parameter is deprecated because `webMethods Broker` is deprecated.

watt.server.publish.validateOnIS

Specifies whether Integration Server validates published documents. Set this parameter to one of the following values:

Specify	To
<code>always</code>	Perform document validation for all published documents. This includes instances of publishable document types for which the Validate when published property is set <code>false</code> .
<code>never</code>	<p>Disable document validation for all publishable document types. This includes instances of publishable document types for which the Validate when published property is set <code>true</code>.</p> <p>Some reasons for disabling document validation include the following:</p> <ul style="list-style-type: none"> ■ You want to improve performance. ■ You want to validate the documents manually. ■ You know that the system that sent Integration Server the data has already validated the data. ■ You prefer to have the messaging provider, rather than Integration Server, validate the documents. ■ Integration Server is sending or receiving native Broker events.

Specify	To
perDoc	<p>Perform document validation only for instances of publishable document types for which document validation is enabled. That is, Integration Server validates published documents that are instances of publishable document types for which the Validate when published property is set <code>true</code>.</p> <p>This is the default behavior.</p>

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

For information about handling native Broker events and specifying validation for an individual publishable document type, see the *Publish-Subscribe Developer's Guide*.

watt.server.recordTodocument.bufferSize

Specifies the size of the buffer that Integration Server uses to write an XML string when converting a document (IData) to an XML string. Increase the buffer size if the majority of the created XML strings will be greater than 4 KB. Decrease the buffer size if the majority of the created XML strings will be small documents less than 4 KB in size. The default size is 4 KB.

watt.server.remoteInvoke.queryCSRFToken

Indicates if, during a remote service invocation, Integration Server queries the remote server for the CSRF token for the current session and then includes the token in the service request. If an Integration Server uses Cross-Site Request Forgery (CSRF) guard, requests sent to the server must include a CSRF token. If the request does not include a CSRF token, the server rejects the request. When an Integration Server performs a remote invoke to execute a service on another Integration Server that uses CSRF guard, the request needs to include the CSRF token. To ensure that the request includes a CSRF token, the requesting Integration Server obtains the CSRF token for the current session from the remote Integration Server. The requesting Integration Server then modifies the request to include the CSRF token. However, this is only necessary if the remote Integration Server uses CSRF guard. Set the `watt.server.remoteInvoke.queryCSRFToken` to `true` if remote Integration Servers use CSRF guard and you want the requesting Integration Server to query for the current CSRF token and include that token in the request. Set to `false` if the remote Integration Servers does not use CSRF guard. The default is `true`.

watt.server.removeDefaultMWSRolesInISACL

Specifies whether Integration Server must remove **My WebMethods Administrator** and **My WebMethods Users** default My webMethods Server (MWS) roles from an ACL. If this property is set to `true` then Integration Server removes the default MWS roles from an ACL. If the property is set to `false` then Integration Server includes the default MWS roles in an ACL. The default value is `false`.

This property was introduced as part of PIE-69462 in IS_10.5_Core_Fix11.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.server.requestCerts

Specifies whether the Integration Server requests a digital certificate from clients that connect to it through SSL. Set this parameter to `true` if you want the server to request certificates. Set it to `false` if you do not want the server to request certificates. The default is `false`.

watt.server.RESTDirective

Specifies an alternative word to use for the `rest` directive in URLs that access resources on Integration Server. By default, this parameter is set as `watt.server.RESTDirective=rest`, which means users must specify the `rest` directive as `rest` (for example, `GET http://host:port/rest/resource/resourceID`). To allow users to specify the `rest` directive as either `rest` or an alternative word, set this parameter to the alternative word. For example, to allow users to specify the `rest` directive as either `rest` or `process`, (`GET http://host:port/rest/resource/resourceID`) or `GET http://host:port/process/resource/resourceID`), set this parameter as `watt.server.RESTDirective=process`.

Important:

- If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.
- When you restart Integration Server after changing the setting of this parameter, all the URL aliases that point to URL paths containing the previous setting of the parameter and created using the Integration Server Administrator will *not* work. Therefore, you must edit the URL path for each of the impacted URL aliases from the Integration Server Administrator to include the updated setting of the parameter. This restriction does not apply to the URL aliases created using Software AG Designer.

watt.server.RESTDirective.V2

Specifies an alternative word to use for the `restv2` directive in URLs that access resources on Integration Server. By default, this parameter is set as `watt.server.RESTDirective.V2=restv2`, which means users must specify the `rest` directive as `restv2` (for example, `GET http://host:port/restv2/resource/resourceID`). To allow users to specify the `rest` directive as either `restv2` or an alternative word, set this parameter to the alternative word. For example, to allow users to specify the directive as either `restv2` or `process`, (`GET http://host:port/restv2/resource/resourceID`) or `GET http://host:port/process/resource/resourceID`), set this parameter as `watt.server.RESTDirective.V2=process`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.response.displayISErrorCode

Indicates whether the HTTP response from Integration Server to a client application in the case of an error situation includes the Integration Server error code in the header and the body of the response.

If you set the value of this parameter to `false`, the HTTP response during error situations includes the error message text without the Integration Server error code in both the header and the body of the response. If you set the value to `true`, the HTTP response during error situations includes the Integration Server error code and the corresponding error message text in the header and body of the response. The default value of the parameter is `true`.

watt.server.rest.addHTTPMethodToInputPipeline

Indicates whether Integration Server adds the *\$httpMethod* variable in the input pipeline with the requested HTTP method while processing REST requests. If this property is set to `true`, Integration Server adds the *\$httpMethod* input variable in the input pipeline. If this property is set to `false`, Integration Server does not add the *\$httpMethod* input variable in the input pipeline. The default value is `false`.

watt.server.rest.includeNullFieldsInResponse

Specifies whether a REST response includes null values for the fields that do not have a value set by the service used as the REST operation. Set the parameter value to `true` to use null values for the empty fields and include them in the REST response. Set the value to `false` to exclude the empty fields from the REST response. The default value is `true`.

Note:

This server configuration parameter is introduced for PIE-72732 in IS_10.5_Core_Fix14.

watt.server.rest.removeInputVariablesFromResponse

Indicates whether Integration Server removes the input variables of a service signature from the pipeline while sending a REST API response. When this property is set to `true`, Integration Server retains only the output variables in the REST API response that matches the response structure as defined in the Swagger document. When this property is set to `false`, Integration Server retains the input variables while sending a REST API response. The default is `true`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.revInvoke.proxyMapUserCerts

For webMethods Enterprise Gateway only. Specifies whether an Enterprise Gateway Server is to perform client authentication itself in addition to passing authentication information to the Internal Server for processing. The default is `false`. If this property is set to `true`, Enterprise Gateway Server will reject all anonymous requests (no certificate and no username/password supplied), even if the request is for an unprotected service on the Internal Server. See [“Configuring webMethods Enterprise Gateway” on page 557](#) for more information.

watt.server.rg.internalregistration.timeout

For webMethods Enterprise Gateway configuration only. Set this value on the Internal Server to specify the time (in seconds) the Internal Server will wait before closing an unresponsive connection to the Enterprise Gateway Server. The default is 0, which means the Internal Server will not time out (a timeout period of infinity).

Important:

Set the value of `watt.server.rg.internalregistration.timeout` on the Internal Server to a value greater than the ping values defined by the `watt.net.socketpool.sweeperInterval` server configuration parameter on the Enterprise Gateway Server. Setting `watt.server.rg.internalregistration.timeout` to a value that is lower than `watt.net.socketpool.sweeperInterval` will cause the Internal Server to close the connection to the Enterprise Gateway Server and re-establish a new connection regularly.

watt.server.rg.internalsocket.timeout

Specifies the length of time, in milliseconds, that Enterprise Gateway Server allows a client request to wait for a connection to the Internal Server before terminating the request with an HTTP 500 Internal Server Error. If a connection to the Internal Server becomes available within the specified timeout period, Enterprise Gateway Server forwards the request to the Internal Server. If a connection does not become available before the timeout elapses, Enterprise Gateway Server returns a HTTP 500-Internal Server Error to the requesting client and writes the following message to the error log:

```
"Enterprise Gateway port {port_number} is unable to forward the request to Internal Server because there are no Internal Server connections available."
```

The default value of this parameter is 0 which means that the client request waits indefinitely for a connection to the Internal Server.

watt.server.scheduler.deleteCompletedTasks

Specifies whether or not scheduled tasks that have expired should be automatically deleted. When set to true, Integration Server deletes expired tasks when they expire. Set this parameter to false if you want Integration Server to wait until the next server restart to delete expired tasks. The default is true.

watt.server.scheduler.ignoreReferenceValidationErrors

Specifies whether Integration Server creates or executes a scheduled task if the scheduled service has a broken reference. Set the property to true to indicate that Integration Server ignores broken references for the scheduled service when executing a scheduled task. Set it to false to indicate that Integration Server should not execute a scheduled task if the scheduled service has one or more broken references. The default is false.

watt.server.scheduler.logical.hostname

Specifies a unique logical name to identify Integration Server instances in a cluster hosted on a single machine. By default, Integration Server uses the host name to identify itself while scheduling tasks. However, when a cluster of Integration Servers are hosted on a single machine, the host name itself cannot uniquely identify the individual Integration Server instances. In such cases, use the `watt.server.scheduler.logical.hostname` property to specify a unique logical name to identify individual Integration Server instances.

The default value for this parameter is the host name.

Keep the following points in mind when setting the `watt.server.scheduler.logical.hostname` property:

- Set this property only if you are running multiple Integration Servers in a cluster on a single machine.
- Set this property on each Integration Server instance in the cluster before scheduling any tasks.
- The logical host name you specify must be unique.
- The logical host name you specify cannot contain a semicolon (;).

This property is also useful when Integration Server is installed on a virtual server whose host name or IP address might change over time. You can use `watt.server.scheduler.logical.hostname` to provide a fixed value that identifies each server in a cluster.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.scheduler.maxWait

Maximum time in seconds Integration Server waits between queries of the task queue. The server periodically checks the queue for tasks that are scheduled to run. If there are no tasks waiting to run, the server waits the maxWait time before checking the queue again. If there are tasks waiting to run, the server checks again at the task's schedule execution time, or after the maxWait time, whichever is earlier. For example, if the pending task is due to execute in 30 seconds and the maxWait time is 60, the server will check the queue again in 30 seconds. The default is 60.

Note:

You must ensure that the value you specify for this parameter must be less than the minimum time interval set for a scheduled task.

If you run a cluster of Integration Servers and schedule a task to run on all servers in the cluster, you might notice tasks starting at different times on the different servers if the servers have different settings for this property. For this reason, if you are running in a clustered environment, all the servers in your cluster should have the same settings for this property. See the *webMethods Integration Server Clustering Guide* for more information about configuring Integration Servers in a cluster.

watt.server.scheduler.threadThrottle

Percentage of Integration Server threads the scheduler process is permitted to use. The default is 75%.

watt.server.securityLog.enableExternalClientIP

Specifies whether the IP address of the external client that invokes a service in Integration Server through Enterprise Gateway is recorded in the security log or not. When set to true, Integration Server records the IP address of the external client in the security log. When set to false, Integration Server records the IP address of Enterprise Gateway in the security log. The default value is false.

Note:

This parameter was introduced for PIE-64558 in IS_10.5_Core_Fix7.

watt.server.securityLog.ignoreXForwardedForHeader

Specifies whether Integration Server ignores the X-Forwarded-For request header and uses the IP address of the proxy server as the client ID in the security log. When set to false, Integration Server obtains the originating client IP address from the X-Forwarded-For request header and uses that client IP address in the security log and security log message. When `watt.server.securityLog.ignoreXForwardedForHeader` is set to true, Integration Server ignores the X-Forwarded-For request header and uses the IP address of the proxy server as the client ID. Note that it is more beneficial for the security log entry to include the IP address of the client that originated the request, especially if the request is an anonymous one. The default is false.

watt.server.serverlogFilesToKeep

Specifies the number of server log files that Integration Server keeps on the file system, including the current server log file. When Integration Server reaches the limit for the number of server log files, Integration Server deletes the oldest archived server log file each time Integration

Server rotates the server log. If you set `watt.server.log.filesToKeep` to 1, Integration Server keeps the current `server.log` file and no previous `server.log` files. When Integration Server rotates the `server.log`, Integration Server does not create an archive file for the previous `server.log`. If you set `watt.server.log.filesToKeep` to 0, or any value less than 1, Integration Server keeps an unlimited number of server log files.

The default value of `watt.server.log.filesToKeep` is -1, indicating that there is no limit to the number of server log files that Integration Server maintains.

For more information about setting the maximum number of server log files that Integration Server keeps, see [“Setting Up the Server Log” on page 213](#).

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.serverlogMaskBearerToken

Specifies whether Integration Server masks the value of a bearer token when writing to the server log. When the server log facility "0038 HTTP Header" is set to Trace and Integration Server receives a request that includes a bearer token, Integration Server writes the Authorization request header to the server log. If `watt.server.serverlogMaskBearerToken` is true, Integration Server writes a string of asterisks to the log rather than the actual bearer token value. If `watt.server.serverlogMaskBearerToken` is false, the bearer token value is written to the log. The default is true.

watt.server.serverlogRotateSize

Specifies the file size at which Integration Server rolls over the `server.log` file. Set this property to N [KB|MB|GB], where N is any valid integer. The minimum size at which Integration Server rotates the `server.log` file is 33KB. If you use KB as the unit of measure, you must set N to a value greater than or equal to 33. If you do not specify a unit of measure, Integration Server treats the supplied N value as bytes. In this case, N must be greater than or equal to 32768 to take effect. Do not include any spaces between the integer and the unit of measure.

There is no default value for this parameter. That is, by default, the parameter has no value. If no value is specified for `watt.server.serverlogRotateSize`, Integration Server rotates the `server.log` file at midnight only.

If an invalid value is specified, Integration Server resets the parameter to -1. Note that a value of -1 results in the same behavior as specifying no value for the parameter.

For more information about the `server.log`, see [“Setting Up the Server Log” on page 213](#).

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.serverlogQueueSize

Controls the number of entries allowed in the server log queue. This property is related to the `watt.server.log.queued` property, which controls whether the server is to write entries directly to the server log, or queue them in memory first and then use a background thread to write them to the server log. If your configuration has the `watt.server.log.queued` property set to true and you notice that expected server log entries are not included in the log, try increasing the queue size. For more information about the server log and the server log queue, see [“Specifying Whether to Queue Server Log Entries” on page 217](#). The default queue size is 8192.

watt.server.serverclassname

This is an internal property. Do not modify.

watt.server.service.blacklist

Specifies, using a comma-separated list or a file, the services on the service blacklist and/or the interfaces whose services are on the service blacklist. Attempts to invoke blacklisted services or services in a blacklisted interface result in an Access Denied error. Any service can be blacklisted, including services delivered with Integration Server such as those in WmPublic and WmRoot. Make sure that a service critical to the functioning of Integration Server is not blacklisted. There is no default value for this server configuration parameter.

Integration Server does not support the use of wildcards with this parameter.

For more information about a service blacklist, see [“Adding Services to a Blacklist” on page 512](#).

Important:

If you change the setting of this parameter or the contents of the file used for the service blacklist, you must restart Integration Server for the changes to take effect.

watt.server.service.lazyEval

Controls whether Integration Server initializes Java services during startup. When set to `true` (the default), Integration Server does not initialize Java Services until they are referenced during runtime. This can increase performance if the server initializes a lot of Java Services at start up.

watt.server.service.list.treatEmptyAsNull

Specifies whether Integration Server assigns a null value to all list data types like Document List, String List, Document Reference List, and Object list during the execution of a flow service without an input value. `True` indicates that the output of the service assigns a null value for the list data type. `False` indicates the output of the service would not assign a null value. The default is `true`.

watt.server.serviceMail

Specifies the email address to which Integration Server sends messages about service failures. There is no default.

When you specify an email address on this parameter, Integration Server creates a simple scheduled task to perform the notification. This task requires a thread, which Integration Server takes from the cronjob-based thread pool. See the `watt.server.cron.maxThreads` and `watt.server.cron.minThreads` parameters for more information about this thread pool.

watt.server.serviceMonitor.queryOnServerId

Specifies monitoring capabilities for the entire stateful cluster and not just for a single instance of the cluster. When set to `true`, then the query includes server ID, and only that server's data are available for monitoring. When set to `false`, the query excludes the server ID, and the entire cluster remains a single entity. The new parameter must be set to the same value on every member of the cluster, and there is no default value for this parameter.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.serviceResults.cache

Specifies the name of the public cache to use for service results caching. If no value is specified for the `watt.server.serviceResults.cache` parameter, Integration Server uses the `ServiceResults` cache in the cache manager specified in the `watt.server.serviceResults.cacheManager` parameter as the service results cache. If the cache manager in the `watt.server.serviceResults.cacheManager` parameter does not contain a cache named `ServiceResults`, Integration Server throws an error at start up and does not cache service results.

Note:

To use the `ServiceResults` system cache located in the `SoftwareAG.IS.Services` system cache manager as the service results cache, do not specify a value for `watt.server.serviceResults.cache` or `watt.server.serviceResults.cacheManager`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.serviceResults.cache.encode

Specifies whether Integration Server encodes service results before serializing the results and placing the results in the service results cache. Set this parameter to `true` if Integration Server encodes service results before serializing the results and placing the data in the service results cache. A value of `true` may prevent a `StackOverflowError` when the service results contain complex data types with many nested levels. However, a value of `true` prevents proper deserialization of results for some data types and might result in empty output. Set this parameter to `false` if Integration Server does not encode service results before serializing the results and placing the data in the service results cache. The default value is `false`.

Important:

You must restart Integration Server for changes to this parameter to take effect.

Note:

The `watt.server.serviceResults.cache.encode` parameter was introduced in `PIE-74072`, which is available in `fix IS_10.5_Core_Fix15` and higher.

watt.server.serviceResults.cacheManager

Specifies the name of the public cache manager that contains the public cache to use for service results caching. If no value is specified for this parameter, Integration Server uses the `SoftwareAG.IS.Services` system cache manager as the service results cache manager. If the `SoftwareAG.IS.Services` system cache does not contain a cache with a name that matches the value of the `watt.server.serviceResults.cache`, Integration Server throws an error at startup and does not cache service results.

Note:

To use the `ServiceResults` system cache located in the `SoftwareAG.IS.Services` system cache manager as the service results cache, do not specify a value for `watt.server.serviceResults.cache` or `watt.server.serviceResults.cacheManager`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.serviceResults.copyOnRead

Specifies how the cache should copy elements that it returns. The default value `true` returns the results by value, while the value `false` returns results by reference.

watt.server.serviceResults.copyOnWrite

Specifies how the cache should copy elements that it writes. The default value `true` returns the results by value, while the value `false` returns results by reference.

watt.server.serviceUsageDSP.RefreshInterval

Specifies the interval, measured in seconds, that elapses before Integration Server Administrator refreshes the data on the **Server > Statistics** page. The default is 90 seconds. The change to the parameter takes effect the next time the **Server > Statistic** page is reloaded.

watt.server.session.locale.ignore

Specifies whether the default locale for the `pub.date*` or `pub.datetime*` services is the server locale or the locale from the session used by the client that invoked the service. When set to `true`, when executing a `pub.date*` service for which a locale input parameter value is not specified, Integration Server uses the server locale as the value of the locale parameter. When set to `false`, when executing a `pub.date*` service or a `pub.datetime*` service for which a `locale` input parameter value is not specified, Integration Server uses the locale from the session used by the client that invoked the service. The default is `false`.

watt.server.session.stateful.enableLimit

Controls whether the stateful session limit feature is enabled. When this property is set to `true`, Integration Server allows stateful sessions to be created as needed until it reaches the maximum allowed, which is specified by the `watt.server.session.stateful.max` property. You can view statistics for stateful sessions on the **Server > Statistics** screen in Integration Server Administrator.

watt.server.session.stateful.max

Specifies the number of concurrent stateful sessions allowed on the Integration Server. If a user attempts to access the server and execute a stateful service while the maximum number of stateful sessions are in use, the server rejects the request and returns an error to the user.

The value for `watt.server.session.stateful.max` must be a positive integer. If a value is not specified or the `watt.server.session.stateful.enableLimit` property is set to `false`, the maximum number of concurrent sessions is defined by your Integration Server license.

Note:

It is recommended that you use the **Maximum Stateful Sessions** field in the Edit Resource Settings page of the Integration Server Administrator to set the maximum value. For more information about setting a maximum limit for stateful sessions, see [“Managing Server Sessions” on page 110](#).

watt.server.session.stateful.warning

Specifies the threshold at which Integration Server starts to warn of insufficient available stateful sessions. When the percentage of available stateful sessions is equal to or less than the value of this property, Integration Server generates a server log message stating:

The default is 25%.

Note:

It is recommended that you use the **Enable Stateful Session Limit** field in the Edit Resource Settings page of the Integration Server Administrator to set the threshold value. For more

information about setting an available stateful sessions warning threshold, see [“Managing Server Sessions”](#) on page 110.

watt.server.setResponse.pre82Mode

Specifies the order in which the `pub.flow:setResponse` service looks for and uses the deprecated input parameters when neither of the input parameters `responseString` or `responseBytes` are provided.

When set to true, the `pub.flow:setResponse` service follows a precedence order similar to what was available in Integration Server 7.1x and 8.0x. Specifically, the service looks for the deprecated parameters in the following order and uses the value of the first parameter that it finds: `response`, `string`, `bytes`

When set to false, the `pub.flow:setResponseservice` follows a precedence order similar to what was available in Integration Server 8.2 and later. Specifically, the service looks for the deprecated parameters in the following order and uses the value of the first parameter that it finds: `string`, `bytes`, `response`

The default is false.

Note:

This parameter is deprecated because the web services implementation introduced in Integration Server 7.1 is deprecated as of Integration Server 10.4. The web services implementation introduced in Integration Server version 7.1 handles web service descriptors that run in pre-8.2 compatibility mode.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.sftp.dateStampFmt

Specifies the format of date to be used in the SFTP client public services, specifically the `pub.client.sftp*` services in the `WmPublic` package. To specify the date format to use, you can use any format that is supported by the Java class `java.text.SimpleDateFormat`. For example, to display the date with the format `08-12-02 14:44:33:1235`, specify `dd-MM-yy HH:mm:ss:SSSS`. The default format for `watt.server.sftp.dateStampFmt` property is `yyyy-MM-dd HH:mm:ss z`.

watt.server.smtpServer

Specifies the SMTP server to use for server error logging and service error email notification. There is no default.

watt.server.smtpServerPort

Specifies the number of the listening port on the SMTP server to which the Integration Server is to send server error logging and service error email notification. The default is 25.

watt.server.smtp.userName

Specifies the username of the account that Integration Server uses to connect to the SMTP server.

watt.server.smtpTransportSecurity

Specifies the type of SSL encryption that Integration Server uses when communicating with an SMTP server.

Set this property to any one of the following values:

When set to	Integration Server
None	<p>Uses a non-secure mode when communicating with an SMTP server.</p> <p>When you specify None, the SMTP server authenticates the SMTP client using only the username and password.</p>
Explicit	<p>Uses explicit security when communicating with an SMTP server. With explicit security, Integration Server establishes an un-encrypted connection to the SMTP server and then upgrades to the secure mode.</p>
Implicit	<p>Uses implicit security when communicating with an SMTP server. With implicit security, Integration Server always establishes an encrypted connection to the SMTP server. Only clients that support SSL will be permitted access.</p>

watt.server.smtpTrustStoreAlias

Specifies the alias for the truststore that contains the list of certificates that Integration Server uses to validate the trust relationship between Integration Server and the SMTP server. If you do not specify a truststore alias, the default truststore alias specified in the `watt.security.trustStoreAlias` property will be used. For more information about truststore alias, see [“Keystore, Truststore, and Key Aliases” on page 480](#).

watt.server.SOAP.addEmptyHeader

Specifies whether Integration Server should create an empty header entry in the SOAP message created by executing the `pub.soap.utils:createSoapData` or `pub.soap.utils:stringToSoapData` services. When this property is set to `true`, Integration Server creates an empty header in the SOAP message. When this property is set to `false`, Integration Server does not create a header entry in the SOAP message. The default is `true`.

The `addEmptyHeader` parameter in the `pub.soap.utils:createSoapData` or `pub.soap.utils:stringToSoapData` services, if set, overrides the `watt.server.SOAP.addEmptyHeader` setting.

For more information about `pub.soap.utils:createSoapData` and `pub.soap.utils:stringToSoapData` services, see *webMethods Integration Server Built-In Services Reference*.

watt.server.SOAP.completeLoad

Specifies whether the XML node added by `pub.utils:addBodyEntry`, `pub.utils:addHeaderEntry`, `pub.utils:addTrailer` will be completely loaded before adding to the SOAP message. When set to `true`, Integration Server loads the entire XML node. When set to `false`, Integration Server does not completely load the XML node before adding it to the service. The default is `true`.

watt.server.soap.convertPlainTextHTTPResponseIntoSOAPFault

Specifies how Integration Server returns the response from a web service invocation when it is acting as a client and the web service results in a plain text HTTP response. This parameter applies only when the web service connector was created using:

- Integration Server version 8.2 or later, but the web service descriptor’s **Pre-8.2 compatibility mode** property is `true`
- A version of Integration Server prior to version 8.2

When the `watt.server.soap.convertPlainTextHTTPResponseIntoSOAPFault` parameter is set to true (the default), Integration Server converts the plain text HTTP response and returns the information from the HTTP response within web service connector's output parameters.

- If the web service connector was created using Integration Server version 8.2 or later and the **Pre-8.2 compatibility mode** property is true, Integration Server returns the converted HTTP payload in the web service connector's `fault/reasons` output parameter.
- If the web service connector was created using a version of Integration Server prior to version 8.2, Integration Server returns the converted HTTP payload in one of the following based on the SOAP protocol in use:
 - `SOAP-FAULT/Fault_1_1/faultstring`
 - `SOAP-FAULT/Fault_1_2/SOAP-ENV:Reason/SOAP-ENV:Text`

Having the plain text HTTP response converted and returned in the web service connector's output parameter can be helpful because it might contain the cause of the exception.

If you set the parameter to false, when the web service results in a plain text HTTP response, Integration Server throws an exception indicating that an invalid SOAP envelope is received.

Note:

This parameter is deprecated because the web services implementation introduced in Integration Server 7.1 is deprecated as of Integration Server 10.4. The web services implementation introduced in Integration Server version 7.1 handles web service descriptors that run in pre-8.2 compatibility mode.

Important:

After updating this parameter, you must restart Integration Server for changes to this parameter to take effect.

watt.server.soap.decodeElementWithPrefix

Specifies whether Integration Server recognizes document types that have defined XML namespace URIs but do not have prefixes associated with each namespace. By default, in Integration Server version 8.2 SP2 and higher, if document types that have a defined XML namespace URI but do not have a prefix associated with each namespace are specified as inputs to services, the SOAP processor fails to recognize the document types at run time. If the `watt.server.soap.decodeElementWithPrefix` property is set to true, Integration Server recognizes the document types that have defined XML namespaces but do not have prefixes associated with each namespace.

The default is false.

The `watt.server.soap.decodeElementWithPrefix` server configuration parameter value affects how Integration Server decodes inbound SOAP messages for web service descriptors created on Integration Server version 7.1.x. Specifically, the parameter affects how Integration Server decodes SOAP response messages received by a 7.1.x web service connector and SOAP request messages received by a 7.1.x web service provider.

- When `watt.server.soap.decodeElementWithPrefix` is set to true, when decoding an inbound SOAP message, Integration Server does not consider namespace and/or prefix declarations when decoding the SOAP message to IData for any web service or web service connector that is part of a web service descriptor created in 7.1.x. Integration Server will map

namespace qualified elements in the XML string to fields in an IS document type even if the field names do not include a prefix. That is, when the parameter is set to true Integration Server uses the decoding behavior available in version 7.1.x. for any web service or web service connector that is part of a web service descriptor created in Integration Server 7.1.x.

Note:

The Integration Server 7.1.x decoding behavior described above is invalid. However, the stricter decoding process in Integration Server 8.2 and later can result in missing or extraneous fields which, in turn, results in validation errors. Setting the `watt.server.soap.decodeElementWithPrefix` to true allows the 7.1.x behavior and does not require changes to the 7.1.x web service descriptors.

- When `watt.server.soap.decodeElementWithPrefix` is set to false, when decoding an inbound SOAP message, Integration Server considers namespace and/or prefix declarations when decoding a SOAP message to IData for any web service or web service connector that is part of a web service descriptor created in Integration Server 7.1.x. Integration Server does not map namespace qualified elements in the XML string to fields in an IS document type if the field names do not include a prefix. That is, when the parameter is set to false, Integration Server uses the decoding behavior available in version 8.2 SP2 and later for any web service or web service connector that is part of a web service descriptor created in Integration Server 7.1.x and later.

Note:

For web service descriptors created in Integration Server 8.2 SP2 onwards, you must associate a prefix with an XML namespace URI for fields in the service signature.

The `watt.server.soap.decodeElementWithPrefix` server configuration parameter also determines whether or not the `pub.xml.xmlNodeToDocument` service considers namespace and/or prefix declarations when decoding namespace qualified elements to corresponding fields in an IS document type.

- When `watt.server.soap.decodeElementWithPrefix` is set to true, the `pub.xml.xmlNodeToDocument` service does not consider namespace and/or prefix declarations. The service will map namespace qualified elements in the XML string to fields in an IS document type even if the field names do not include a prefix. Note that this behavior is considered invalid but is allowed to provide backward compatibility with services developed on earlier versions of Integration Server.
- When `watt.server.soap.decodeElementWithPrefix` is set to false, the `pub.xml.xmlNodeToDocument` service considers namespace and/or prefix declarations. The service does not map namespace qualified elements in the XML input to fields in an IS document type if the field names do not include a prefix. The `pub.xml.xmlNodeToDocument` service does not populate non-prefixed fields. Instead the service adds new fields that use the `prefix:name` structure and populates those fields with values from the XML instance.

Important:

After updating the `watt.server.soap.decodeElementWithPrefix` parameter, you must restart Integration Server for changes to the parameter to take effect.

watt.server.SOAP.default.endpointHTTP

Specifies the default provider web service endpoint alias for the HTTP protocol. If the value of this parameter is empty, there is no default provider web service endpoint alias for HTTP.

There is no default value for this parameter.

The `watt.server.SOAP.default.endpointHTTP` parameter displays the value specified in the **HTTP** parameter on the **Settings > Web Services > Set Default Provider Endpoint Aliases** screen in Integration Server Administrator. Software AG recommends that you specify the default provider endpoint alias using the **HTTP** parameter on the **Settings > Web Services > Set Default Provider Endpoint Aliases** screen instead of by changing the value of the `watt.server.SOAP.default.endpointHTTP` parameter. For more information about setting a default provider endpoint alias, see [“Setting a Default Endpoint Alias for Provider Web Service Descriptors” on page 353](#).

watt.server.SOAP.default.endpointHTTPS

Specifies the default provider web service endpoint alias for the HTTPS protocol. If the value of this parameter is empty, there is no default provider web service endpoint alias for HTTPS.

There is no default value for this parameter.

The `watt.server.SOAP.default.endpointHTTPS` parameter displays the value specified in the **HTTPS** parameter on the **Settings > Web Services > Set Default Provider Endpoint Aliases** screen in Integration Server Administrator. Software AG recommends that you specify the default provider endpoint alias using the **HTTPS** parameter on the **Settings > Web Services > Set Default Provider Endpoint Aliases** screen instead of by changing the value of the `watt.server.SOAP.default.endpointHTTPS` parameter. For more information about setting a default provider endpoint alias, see [“Setting a Default Endpoint Alias for Provider Web Service Descriptors” on page 353](#).

watt.server.SOAP.defaultProtocol

Specifies the default protocol that Integration Server uses for new SOAP messages. Specify SOAP 1.1 Protocol or SOAP 1.2 Protocol. The default is SOAP 1.1 Protocol.

watt.server.SOAP.directive

Specifies a different word to use for the SOAP directive in URLs that route requests to the Integration Server SOAP handler. By default, this parameter is set as

`watt.server.SOAP.directory=soap`, which means users must specify the SOAP directive as `soap` (`http://host:port/soap`). To allow users to specify the SOAP directive as a different word instead, set this parameter to that word. For example, to allow users to specify the SOAP directive as `endpoint`, (`http://host:port/endpoint`), set this parameter as

`watt.server.SOAP.directive=endpoint`.

watt.server.SOAP.encodeXSIType

Indicates where the `xsi:type` will appear as an attribute for an element in a SOAP message that specifies RPC/Encoded. A value of `true` indicates that Integration Server includes the `xsi:type` attribute for an element. `false` indicates that the `xsi:type` attribute will be omitted. The default is `true`. Software AG recommends using a value of `true` for this parameter.

watt.server.SOAP.encodeXSITypeValue

Indicates whether Integration Server should include the `xsi:type` attribute and its value for an element of `xsd:anyType` in SOAP requests and responses. When set to `false`, Integration Server omits the `xsi:type` attribute and its value for an element of `xsd:anyType` in SOAP requests and responses. When set to `true`, Integration Server includes the `xsi:type` attribute and its value for an element of `xsd:anyType` in SOAP requests and responses. The default is `true`.

Note:

The `watt.server.SOAP.encodeXSITypeValue` server configuration parameter affects only those SOAP requests and responses created by Integration Server.

watt.server.SOAP.generateNilTags

Specifies whether or not Integration Server generates an `xsi:nil` attribute for an element in a SOAP message. When set to true, Integration Server generates an `xsi:nil` attribute for an element that is nillable (the **Allow null** property for the corresponding field is set to true) and the field is null at run time. When `watt.server.SOAP.generateNilTags` is set to false, Integration Server omits the `xsi:nil` attribute for an element even if the corresponding field is nillable and the field is null at run time. The default is true.

watt.server.SOAP.generateRequiredTags

Specifies whether or not a SOAP message generated by Integration Server includes empty element tags for required parameters for which a value was not supplied at run time. When set to true, Integration Server generates empty element tags for required parameters for which a value is not specified. When `watt.server.SOAP.generateRequiredTags` is set to false, Integration Server omits empty element tags for required parameters for which a value is not specified. The default is true.

watt.server.SOAP.hideEPRHostInFault

Hides the endpoint reference host name and IP address details in the SOAP fault. When this parameter is set to true, Integration Server replaces the host name and IP address with `*.*` in the SOAP fault. When this parameter is set to false, Integration Server includes the host name and IP address details of the endpoint reference in the SOAP fault. The default is false.

Note:

This parameter applies only when the **Pre-8.2 compatibility mode** property of the web service descriptor is set to false.

watt.server.SOAP.hideFaultReason

This parameter enables you to remove fault details from responses to Web services requests. By default, this parameter is set to false and responses include fault details. If the responses include fault details, client applications may spoof or modify the actual content on a web page. To prevent content spoofing, set this parameter to true.

Important:

You must also set `watt.server.http.returnValueException` to false or `webMethods` to prevent Integration Server from returning the stack trace to clients in case a Web service returns an error.

watt.server.SOAP.identifyIsGeneratedWSDL

When creating or refreshing a web service descriptor from a WSDL document, detects whether the WSDL document was originated by Integration Server. When `watt.server.SOAP.identifyIsGeneratedWSDL` property is set to true, Integration Server detects whether the WSDL document was generated by Integration Server and if so, removes the top-level elements in the input and output message. When set to false, Integration Server processes the WSDL document as written and does not remove the top-level elements in the input and output message when creating the web service descriptor. The default value is true.

This parameter affects WSDL first provider web service descriptors and consumer web service descriptors.

Important:

Changing the value of `watt.server.SOAP.identifyISGeneratedWSDL` affects the logic for creating and refreshing web service descriptors. If you change the parameter value and then refresh a web service descriptor that was created before the parameter value changed, the refreshed web service descriptor will be different from the original web service descriptor.

watt.server.SOAP.ignoreMissingResponseHeader

Determines whether missing required headers in a SOAP response, including a SOAP fault, results in an error. When set to `true`, a SOAP response that does not contain a required header will not result in an error. When set to `false`, a SOAP response that does not contain a required header results in an error. The default is `false`.

Note that all headers in a web service descriptor, whether generated from the original WSDL document or added to the web service descriptor, are treated as required. This is an application of the WS-I basic profile rules declaring that all headers in a WSDL be treated as required. Prior to Integration Server version 9.0, when processing a SOAP response received by a web service descriptor, Integration Server did not properly validate that all of the SOAP headers were present. As a result, Integration Server did not throw an error when a SOAP response was missing a SOAP header. Beginning in Integration Server version 9.0, Integration Server properly validates SOAP responses for required headers. If a required header is not present, Integration Server throws the following error: “[ISS.0088.9443] One or more required Headers <headerName> are not present in the SOAP response.” While failure when a required header is missing is the correct behavior, Integration Server provides a configuration property to control whether missing required headers in a SOAP response results in an error. This can be useful in migration situations.

watt.server.SOAP.inbound.CDATA.removeTags

Specifies whether Integration Server removes or preserves the CDATA delimiter tags found in an inbound SOAP request. When set to `true`, during inbound processing, Integration Server removes the initial CDATA tag “<![CDATA[” and the terminating tag “]]>” from the string values passed as input to the target web service. When set to `false`, Integration Server preserves the CDATA delimiter tags in inbound SOAP requests; the CDATA delimiter tags will remain in the text of the request and reach the target web service. The default value is `true`.

watt.server.SOAP.MTOMStreaming.cachedFiles.location

Specifies the absolute path to the hard disk drive space that Integration Server uses to temporarily store inbound SOAP messages when performing MTOM streaming. This parameter takes effect only when MTOM streaming is enabled (i.e., `watt.server.SOAP.MTOMStreaming.enable` is set to `true`).

You can specify a directory on the same machine as the Integration Server or in any other location accessible to Integration Server, such as a mapped logical drive or network directory. The default value is *Integration Server_directory* \instances*instance_name*\temp\mtom\cached files.

Note:

If your Integration Server is in a clustered environment, you must set this property in all the servers in the cluster. See *webMethods Integration Server Clustering Guide* for more information about configuring Integration Server in a cluster.

watt.server.SOAP.MTOMStreaming.enable

Indicates whether MTOM streaming is enabled for inbound SOAP messages and whether HTTP chunking is enabled for outbound requests. Set this property to true to enable MTOM streaming for inbound SOAP messages and HTTP chunking for outbound requests. Set this property to false to disable MTOM streaming and HTTP chunking. The default is `false`.

watt.server.SOAP.MTOMStreaming.threshold

Specifies the maximum number of bytes for Integration Server to handle an inbound MTOM attachment field as an in-memory `ByteStream`. Integration Server writes an inbound MTOM attachment field exceeding this size to a temporary disk file and processes the inbound MTOM Streams as a `FileStream`. This parameter takes effect only when MTOM streaming is enabled (`watt.server.SOAP.MTOMStreaming.enable` is set to true). The default is 4000 bytes.

The `watt.server.SOAP.MTOMStreaming.cachedFiles.location` value determines the location of the temporary disk files.

watt.server.SOAP.MTOMThreshold

Specifies the field size, in kilobytes, that determines whether Integration Server handles base64binary encoded data in an outbound SOAP request as a MIME attachment or whether it sends it inline in the SOAP message. If the web service descriptor for the SOAP message enables attachments for the SOAP request, Integration Server passes as MIME attachments any base64 fields in a SOAP message that are larger than the threshold. The default is 0.

watt.server.SOAP.pre82WSD.ignoreVersionMismatch

For a web service provider that was created in an Integration Server release earlier than 8.2.2 and for which the **Pre-8.2 compatibility mode** property is set to true, specifies whether to emulate pre-8.2 behavior for process SOAP requests. In Integration Server 7.1.3, Integration Server allowed web service providers to process SOAP requests with a SOAP version that did not match the SOAP version that was declared in the WSDL binding. When set to true, this property provides backward compatibility so that consumers do not have to update their SOAP requests.

When set to false, Integration Server uses stricter validation that does not allow processing when this mismatch exists. The default value is `false`.

Note:

This parameter is deprecated because the web services implementation introduced in Integration Server 7.1 is deprecated as of Integration Server 10.4. The web services implementation introduced in Integration Server version 7.1 handles web service descriptors that run in pre-8.2 compatibility mode.

watt.server.SOAP.preserveCDATA

Specifies whether Integration Server encodes CDATA blocks in outbound messages. When set to true, when Integration Server encounters a CDATA in an outbound SOAP message, Integration Server will maintain it in the wire request unchanged and unencoded. When set to false, Integration Server treats the CDATA section as regular text resulting in the html encoding of the CDATA tag and illegal characters in the CDATA section. Note that encoding occurs before Integration Server places the outbound SOAP message on the wire. Outbound SOAP messages that are affected by this setting include SOAP requests sent by web service connectors and SOAP responses sent by web service providers. The default value is true, CDATA is preserved and not encoded.

watt.server.SOAP.request.timeout

Specifies the length of time, in seconds, Integration Server will wait for the SOAP response from the server hosting the remote procedure. If Integration Server does not receive a response in the allotted time, it terminates the request. The default value is -1, which indicates that Integration Server will use the value set for the `watt.net.timeout` property.

watt.server.SOAP.retainUndeclaredNamespace

Specifies whether Integration Server retains namespaces from an `xsd:any` element when decoding a SOAP request or SOAP response. Integration Server considers the namespaces from an `xsd:any` element to be undeclared because the possible elements and corresponding namespaces are not defined in a corresponding document type. When set to `true`, Integration Server preserves namespace declarations for the `xsd:any` element. Additionally, the corresponding field for an element that belongs to the namespace includes the prefix in the field name. When set to `false`, Integration Server does not retain the namespace declarations for an `xsd:any` element and the name of the field corresponding to an element in the declared namespace names do not include the prefix. The default is `true`.

When this parameter is set to `true`, Integration Server preserves namespace declarations by inserting the following field into the document for each namespace declaration in the `xsd:any` element:

```
@xmlns:<prefix>
```

Where `<prefix>` is the prefix defined in the SOAP message. If no prefix is defined, the variable name will be `@xmlns`.

Note:

Note that the `watt.server.SOAP.retainUndeclaredNamespace` value affects decoding of all SOAP messages for web services.

watt.server.SOAP.setNamespaceURIsToRoot

Specifies how Integration Server declares XML namespaces in a SOAP response. When the `watt.server.SOAP.setNamespaceURIsToRoot` property is set to `true`, Integration Server declares the namespace and prefix once at the root element of the SOAP response and then uses the prefix for each element in the response. When set to `false`, Integration Server declares the namespace the way it is defined in the original document, either defining each element explicitly with the full namespace or declaring the namespace and prefix just once at the root element. The default value for this property is `false`.

watt.server.SOAP.streamHandlers

Specifies the custom SOAP handlers that expect stream inputs. Enter the SOAP handlers as a semi-colon (;) separated list. There is no default.

watt.server.SOAP.treatNilAsNull

Indicates whether or not Integration Server, when decoding a SOAP message, generates an `@xsi:nil` field for an element for which the `xsi:nil` attribute is set to `true`. When set to `true`, Integration Server treats an element that carries the `xsi:nil` attribute as a null value and does not create an `@xsi:nil` attribute for the element. When this parameter is set to `false`, Integration Server generates an `@xsi:nil` field for an element that carries the `xsi:nil=true` attribute. This configuration parameter affects all web services running on Integration Server. The default is `true`.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.SOAP.useMultiReference

For elements that appear in multiple places in a SOAP message, specifies whether Integration Server serializes each occurrence of the element or serializes it in only one place and uses the href attribute in the other locations. When set to `true`, Integration Server serializes one location and uses the href attribute in other locations. This parameter applies to RPC/Encoded web services only. The default is `true`.

watt.server.SOAP.useStringforAnyTypewithSimpleValue

Specifies how Integration Server decodes an `xsd:anyType` element with simple content in SOAP responses. When set to `false`, Integration Server decodes `xsd:anyType` elements with simple content into an `IData` with an `@xsi:type` field to retain the `xsi:type` value and a `*body` field to contain the element value. This is the default behavior.

When the `watt.server.SOAP.useStringforAnyTypewithSimpleValue` server configuration parameter is set to `true`, Integration Server decodes `xsd:anyType` elements with simple content into a `String` type field that contains the element value. The element's `xsi:type` information will be gone. You do not have to restart Integration Server for the change to take effect.

watt.server.soap.validateInput

Specifies whether or not a validation error occurs when an inbound SOAP request includes fields that are not declared in the service input signature. Set this parameter to `true` if Integration Server returns a validation error when the body of an inbound SOAP request includes field that are not declared in the service input signature. Set this parameter to `false` if Integration Server continues with processing an inbound SOAP request even if the SOAP body includes fields that do not exist in the service input signature. The default is `true`.

watt.server.soap.validateResponse

Enables or disables SOAP response validation. When set to `true`, Integration Server validates the SOAP response received by a web service connector. When set to `false`, Integration Server does not validate the received SOAP response. The default is `true`.

watt.server.SOAP.validateSOAPMessage

When Integration Server acts as the web service provider, indicates whether Integration Server validates inbound SOAP requests and outbound SOAP responses. When set to `true`, Integration Server validates inbound SOAP requests and outbound SOAP responses against the SOAP schema. Be aware that the validation process checks only that the message envelope is structured correctly. For example, Integration Server checks that the message has at least one body element and there is at most one header element. Integration Server does not validate any of the data carried by the message. The default is `true`.

If you are operating in a production environment where the validity of the messages submitted to the server is assured, you might set `watt.server.SOAP.validateSOAPMessage` to `false` to optimize performance.

watt.server.SOAP.warnOnPartValidation

When creating a web service descriptor from a WSDL document, indicates whether Integration Server should treat message parts that are defined by the type attribute instead of the element attribute as a warning and not an error. Set this parameter to `true` to indicate that Integration Server should return a warning and allow the web service descriptor to be created. Set this

parameter to `false` to indicate that Integration Server should return an error and not allow the web service descriptor to be created. The default is `false`.

watt.server.soapJMS.defaultMessageType

Specifies the default message type for web service request messages sent using SOAP over JMS. Set this parameter to "BytesMessage" to send a message whose body contains a stream of uninterrupted bytes. Set this parameter to "TextMessage" to send a message whose body contains a Java string. BytesMessage is considered to be a more efficient way of sending JMS messages. However, you may want to set the parameter to TextMessage for debugging purposes because the resulting messages will be in a human-readable format. Keep in mind that the message type of the request message determines the message type of the response message. The default is BytesMessage.

Important:

You must restart Integration Server for changes to this parameter to take effect.

Note:

The default message type can be overwritten during web service connector execution by setting the `jms.messageType` property in the `transportHeaders` input parameter.

Note:

If you set this parameter to TextMessage, Integration Server sends all web service responses as TextMessage. If the request message is a BytesMessage and `watt.server.soapJMS.defaultMessageType` is set to TextMessage, Integration Server overrides the request message type and sends the response as a TextMessage. This can be useful in debugging situations.

watt.server.soapjms.request.timeout

Specifies the number of seconds that Integration Server waits for a response to a SOAP request sent using SOAP over JMS. This value must be an integer greater than or equal to zero. A value of 0 indicates that Integration Server will wait indefinitely for a response. The default is 10 seconds.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.SoapRPC.checkHeaders

Indicates whether Integration Server should check SOAP headers for SOAP RPC requests. The default is `true`.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.SoapRPC.distinguishDuplicateElements

Indicates whether Integration Server should differentiate identically named arrays in the SOAP response for a SOAP/RPC web service. When set to `true`, Integration Server appends a number to the `xsi:type` value to distinguish between identically named array elements in the SOAP response, for example `elementName` and `elementName1`. When set to `false`, Integration Server does not differentiate identically named arrays in the SOAP response. The default is `true`.

watt.server.SoapRPC.useSecondaryType

Instructs Integration Server to use a second type definition when creating the SOAP response for a service whose input or output signatures contain identically named variables of different types. When creating a WSDL from a provider web service descriptor that contains a service with identically named fields of different types, Integration Server renames the second instance of the field type in the WSDL. At run time, for RPC-Encoded SOAP binding, Integration Server encodes the types in the SOAP response. When this property is set to `true`, the SOAP response refers to the renamed type definition. When set to `false`, the SOAP response refers to the original type definition instead of the renamed one. The default is `false`. This property is applicable to RPC-Encoded SOAP binding only.

watt.server.ssl.keyStoreAlias

Name of the keystore alias for the Integration Server keystore that contains the information needed to establish an SSL connection with an SSL-enabled external server. There is no default value for this parameter. For more information about storing keystore information for an SSL connection to an external server, see [“Storing SSL Information for the Integration Server JVM in a Secure Manner” on page 464](#).

Important:

You must restart Integration Server for changes to this parameter to take effect.

Important:

You must also restart Integration Server if you modify the contents of the keystore whose alias is used as the value of the `watt.server.ssl.keyStoreAlias` parameter. Changes made to the keystore will take effect after the restart.

watt.server.ssl.trustStoreAlias

Name of the truststore alias for the Integration Server truststore that contains the information needed to establish an SSL connection with an SSL-enabled external server. There is no default value for this parameter. For more information about storing truststore information for an SSL connection to an external server, see [“Storing SSL Information for the Integration Server JVM in a Secure Manner” on page 464](#).

Important:

You must restart Integration Server for changes to this parameter to take effect.

Important:

You must also restart Integration Server if you modify the contents of the truststore whose alias is used as the value of the `watt.server.ssl.trustStoreAlias` parameter. Changes made to the truststore will take effect after the restart.

watt.server.stats.logfile

Specifies the name of the file to receive statistics. The file contains the data that appears on the **Server > Statistics** page. The default file name is `logs\stats.log`.

watt.server.stats.logFilesToKeep

Specifies the number of `stats.log` files that Integration Server keeps on the file system, including the current log file. When Integration Server reaches the limit for the number of `stats.log` files, each time Integration Server rotates the `stats.log`, Integration Server deletes the oldest archived `stats.log` file. If you set `watt.server.stats.logFilesToKeep` to 1, Integration Server keeps the current `stats.log` file and no previous `stats.log` files. That is, when Integration Server rotates the `stats.log`, Integration Server does not create an archive file for the previous `stats.log`. If you set

watt.server.stats.logFilesToKeep to 0, or any value less than 1, Integration Server keeps an unlimited number of stats.log files. The default value of watt.server.stats.log.filesToKeep is 0.

If you reduce the number of stats.log files that Integration Server keeps and then restart Integration Server, the existing logs will not be pruned until Integration Server rotates the stats.log.

Integration Server uses the following naming format for the archived stats.log files: stats.yymmdd_hhmmss.log where the timestamp indicates the time at which Integration Server archived the log file

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.stats.logRotateSize

Specifies the file size at which Integration Server rolls over the stats.log. Set this property to N[KB|MB|GB], where N is any valid integer. The minimum size at which Integration Server rotates a stats.log is 33KB. If you use KB as the unit of measure, you must set N to a value greater than or equal to 33. If you do not specify a unit of measure, Integration Server treats the supplied N value as bytes. In this case, N must be greater than or equal to 32768 to take effect. Do not include any spaces between the integer and the unit of measure. If an invalid value is specified, Integration Server proceeds as if no value was specified for the parameter. If you set the value using the **Extended Settings** page in Integration Server Administrator, validation prevents an invalid value from being saved.

There is no default value for this parameter. That is, by default, the parameter has no value. If no value is specified for watt.server.stats.logRotateSize, Integration Server rotates the stats.log at the interval specified by watt.server.statsLogRotateInterval only.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.stats.pollTime

Specifies the number of seconds between updates of statistics loggings. For a stand-alone Integration Server, the watt.server.stats.pollTime must be an integer greater than or equal to 0 (zero). For an Integration Server in a cluster, the watt.server.stats.pollTime must be an integer greater than 0 (zero) but less than or equal to 60. The default is 60.

Integration Server Administrator displays this value as the interval of the Statistics Log task on the **Server > Scheduler > View System Tasks** page.

watt.server.statsLogRotateInterval

Specifies the length of the log recycle interval (in minutes) for the stats.log file. The recycle interval is the period of time that Integration Server retains statistics log records. The default is 1440 (24 hours).

Integration Server Administrator displays the watt.server.statsLogRotateInterval value as the interval for the Stats Log Recycle task on the **Server > Scheduler > View System Tasks** page. Integration Server Administrator displays this value in seconds instead of minutes. For example, if you set watt.server.statsLogRotateInterval to the default value of 1440, Integration Server Administrator displays the value for the Log Recycle interval as 86400 seconds. For more information about viewing system tasks, see [“Viewing Scheduled User Tasks” on page 721](#).

After Integration Server startup, Integration Server immediately begins rotating the stats.log at the time specified by the watt.server.statsLogRotateInterval. If you set

`watt.server.statsLogRotateInterval` to 2 minutes and Integration Server started at 9:30 pm, Integration Server rotates the `stats.log` at 9:32, 9:34, 9:36, and so forth. An exception to this behavior is that when the log recycle time is greater than the time from Integration Server startup to midnight, Integration Server first rotates the `stats.log` at midnight. Thereafter, Integration Server rotates the log at the specified interval. For example, if the `watt.server.statsLogRotateInterval` is set to 300 minutes (5 hours) and Integration Server starts up at 9:30 pm, Integration Server rotates the `stats.log` file at midnight and thereafter rotates the `stats.log` every 5 hours

Note:

The `watt.server.statsLogRotateInterval` server configuration parameter was previously named the `watt.server.logRotateInterval`.

watt.server.storage.addKeyToStoreIfNotPresent

Specifies whether the `pub.storage:lock` service adds the specified key to the data store if the key does not exist in the data store at the time the service executes. Set this parameter to `true` to have the `pub.storage:lock` service add the specified key to the data store if the key does not exist at the time the service executes. The `pub.storage:lock` service creates the specified key, assigns it a `NULL` value, and then locks the entry in the data store. Set this parameter to `false` if you do not want the `pub.storage:lock` service to add the key to the data store. The `pub.storage:lock` service is a `NOP` (no operation) if the specified key value does not exist in the data store. The default value is `false`. For more information about the `pub.storage` services, see the *webMethods Integration Server Built-In Services Reference*.

watt.server.storage.lock.maxDuration

Specifies the maximum number of milliseconds a `pub.storage` service will hold a lock. The default value is 180000 milliseconds (3 minutes). For more information about the `pub.storage` services, see the *webMethods Integration Server Built-In Services Reference*.

watt.server.storage.lock.maxWait

Specifies the maximum number of milliseconds a `pub.storage` service will wait to obtain a lock. The default value is 240000 milliseconds (4 minutes). For more information about the `pub.storage` services, see the *webMethods Integration Server Built-In Services Reference*.

watt.server.storage.lock.sweepInterval

Specifies how often (in seconds) Integration Server deletes expired `pub.storage` locks. An expired `pub.storage` lock is one that is older than the value specified on the `watt.server.storage.lock.maxDuration` parameter. The default is 5 seconds. If the `watt.server.storage.lock.sweepInterval` parameter is set to a value less than one, Integration Server ignores the setting and uses 5 seconds instead. Integration Server does not have to be restarted for changes to this property to take effect. For more information about the `pub.storage` services, see the *webMethods Integration Server Built-In Services Reference*.

watt.server.strictAccessExceptionLogging

Specifies whether Integration Server will log HTTP 401 Access Denied as an error and trigger a notification. When this property is set to `true`, Integration Server will log HTTP 401 Access Denied as an error and trigger notifications. When this property is set to `false`, Integration Server will not log HTTP 401 Access Denied as an error and will not trigger a notification. The default is `false`.

watt.server.suppresswarn

Specifies whether, when running a C service in Designer, Integration Server should write the warning messages issued about the missing variables to the server log. When this property is set to `true`, Integration Server does not write these messages to the server log. When this property is set to `false`, Integration Server writes messages about missing variables to the server log. The default is `false`.

watt.server.sync.timeout

Specifies the time period that a lock object exists for a given key for which a notification has been issued. After calling a `pub.sync.notify` service to create the notification, a `pub.sync.wait` can receive the notification. Any thread that is waiting for a notification for the *key*, receives the notification as long as the lifespan of the wait request overlaps with the lifespan of the notification. However, if a thread with an exclusive wait is registered for the notification *key* before any other service is waiting, the notification ends as soon as the exclusive wait receives the notification. The default is 60 seconds.

watt.server.systemtasks.debug

Enables debug logs related to system tasks. By default its value is `false`. To enable the logging, change the value to `true`. The logs are written to `server.log` under the logging facility **User Task Scheduler**. The system task logs include information such as when a task got created, when it got terminated, and for recurring tasks, when it will run again.

watt.server.thread.aging.limit

Specifies the length of time, in minutes, a thread can remain in the thread pool. When this length of time is reached and the thread has been used the maximum number of times as specified by the `watt.server.thread.usage.limit` parameter, Integration Server waits until the service is complete and releases the thread from the pool. A value of -1 indicates that threads can remain indefinitely in the thread pool. The default is 60 minutes.

Note:Integration Server releases an expired thread only when the values for both `watt.server.thread.aging.limit` and `watt.server.thread.usage.limit` are reached. Both parameters must be set to a positive integer. If either or both parameters have a value of -1, both parameters will be disabled.

Important:

Use this setting with extreme care because it will affect server performance and throughput.

watt.server.thread.usage.limit

Specifies the maximum number of times a pooled thread can be used. When this maximum number is reached and the length of time specified by the `watt.server.thread.aging.limit` parameter is reached, Integration Server waits until the service is complete and releases the thread from the pool. A value of -1 indicates that threads can be reused indefinitely. The default is 1000.

Note:Integration Server releases a used thread only when the values for both `watt.server.thread.aging.limit` and `watt.server.thread.usage.limit` are reached. Both parameters must be set to a positive integer. If either or both parameters have a value of -1, both parameters will be disabled.

Important:

Use this setting with extreme care because it will affect server performance and throughput.

watt.server.threadKill.enabled

Controls whether the thread kill facility is enabled. This facility allows you to cancel or kill service threads in cases where a service is unresponsive. When you cancel a thread, Integration Server frees up resources that are held by the thread. When you kill a thread, Integration Server stops the thread, releases it from the thread pool, and replenishes the thread pool. The default is true. When the thread kill facility is enabled, you can cancel or kill threads by going to the **Server > Statistics > System Threads** screen. For more information about canceling and killing threads, refer to [“Canceling and Killing Threads Associated with a Service” on page 708](#).

watt.server.threadKill.interruptThread.enabled

Specifies whether Integration Server should interrupt a service for which a timeout is triggered or is cancelled. If set to true, when a timeout is triggered or the service is cancelled, Integration Server interrupts the service thread that is executing a service. The default is false.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.server.threadKill.timeout.enabled

Controls how Integration Server handles the timeout setting of flow steps. Flow steps have a timeout property that controls how long the step can run. By default (true), when the step has exceeded the timeout period, Integration Server raises a FlowTimeoutException, and flow execution continues with the next step. When this property is set to false, it is possible for the flow step to run beyond its timeout period, in some cases. For instance, if a flow step calls another service, for example ServiceA, and ServiceA is executing when the timeout period has passed, ServiceA will continue executing past the timeout period. When ServiceA ends, it passes control back to the parent flow service, which then issues an exception.

watt.server.threadPool

Specifies the maximum number of threads that the server maintains in the thread pool that it uses to run services. If this maximum number is reached, the server waits until services complete and return threads to the pool before running more services. The default is 75.

watt.server.threadPool.cloudRequests

Specifies the maximum percentage of the server thread pool that can be used for processing Integration Cloud requests. The default is 5.

Note:

The resolution to PIE-60686 (IS_10.5_WmCloud_Fix1 and higher) changed the default from 75 to 5.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.threadPoolMin

Specifies the minimum number of threads that the server maintains in the thread pool that it uses to run services. When the server starts, the thread pool initially contains this minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed, which is specified by the `watt.server.threadPool` setting. The default is 10.

watt.server.transaction.recovery.abandonTimeout

If an error occurs while Integration Server tries to resolve an uncompleted XA transaction, specifies the maximum length of time (in minutes) during which Integration Server should make additional attempts. The default is 5 minutes.

watt.server.transaction.recovery.sleepInterval

If an error occurs while Integration Server tries to resolve an uncompleted XA transaction, specifies the length of time (in seconds) that Integration Server waits between additional attempts. The default is 30 seconds.

watt.server.transaction.xastore.maxTxnPerFile

Specifies the maximum number of unique XA transactions in an XA recovery log file. When the XA recovery log file reaches the maximum number of transactions, Integration Server creates a new file. The default is 2000 transactions.

Consider increasing the maximum number of unique XA transactions for the XA recovery log file if there are more than 2000 active XA transactions and Integration Server exhibits a performance delay due to input/output. Increasing the number of unique XA transactions allowed per file decreases the number of files used for the XA recovery log, which, in turn, may result in fewer files for Integration Server to search when performing XA recovery.

Decreasing the number of unique XA transactions stored in file may help during transaction recovery as it might decrease the time to read and consolidate open transactions.

watt.server.transaction.xastore.performXALogging

Specifies whether Integration Server writes transaction information to the XA recovery store. Set to `true` to instruct Integration Server to log information about the state and progress of each XA transaction. Set to `false` to instruct Integration Server to skip logging XA transaction information. The default is `true`.

Important:

If you set this property to `false`, Integration Server does not log any information to the XA recovery store while processing a transaction, making transaction recovery impossible. If you want Integration Server to automatically resolve incomplete transactions or you want to manually resolve incomplete transactions, Integration Server must perform XA logging.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.trigger.interruptRetryOnShutdown

Specifies whether or not a request to shut down the Integration Server interrupts the retry process for a webMethods messaging trigger service. If this parameter is set to `false`, Integration Server waits for the maximum retry attempts to be made before shutting down. Integration Server will also shut down if the trigger service executes successfully during a retry attempt. If this parameter is set to `true`, Integration Server waits for the current service retry to complete. If the webMethods messaging trigger service needs to be retried again (the service ends because of an `ISRuntimeException`), the Integration Server stops the retry process and shuts down. Upon restart, the transport (Broker, Universal Messaging, or, for a local publish, the transient store) redelivers the document to the trigger for processing. The default is `false`.

watt.server.trigger.keepAsBrokerEvent

Specifies whether Integration Server should bypass decoding that is normally performed when documents are retrieved from the Broker on behalf of a webMethods messaging trigger. If this

property is set to `true`, Integration Server passes the value of the Broker event to the trigger service in an object called `$brokerEvent` and no decoding is performed. Set this parameter to `true` if Integration Server is receiving native Broker events. The default is `false`.

For more information about publishing native Broker events, see the *Publish-Subscribe Developer's Guide*.

Note:

This parameter is deprecated because `webMethods Broker` is deprecated.

watt.server.trigger.local.checkTTL

Specifies whether Integration Server should strictly enforce a locally published document's time-to-live. When this parameter is set to `true`, before processing a locally published document in a trigger queue, Integration Server determines whether the document has expired. Integration Server discards the document if it has expired. The default is `false`.

watt.server.trigger.managementUI.excludeList

Specifies a comma-delimited list of triggers to exclude from the **webMethods Messaging Trigger Management** pages in Integration Server Administrator. Integration Server also excludes these `webMethods` messaging triggers from trigger management changes that suspend or resume document retrieval or document processing for all triggers. Integration Server does *not* exclude these `webMethods` messaging triggers from changes to capacity, refill level, or maximum execution threads that are made using the global trigger controls (Queue Capacity Throttle and Trigger Execution Threads Throttle).

You can specify the fully qualified names of all the `webMethods` messaging triggers that you want to exclude. You can also use pattern matching to exclude a group of triggers by specifying the beginning portion of the fully qualified name and following it with an asterisk (*). The Integration Server excludes all triggers that begin with the supplied pattern. For example, if you want to exclude all triggers located in the `pub.prt` folder, specify:

```
watt.server.trigger.managementUI.excludeList = pub.prt*
```

Specifies the interval, measured in seconds, at which Integration Server executes resource monitoring services for `webMethods` messaging triggers. A resource monitoring service is a service that you create to check the availability of resources used by a trigger service. When it suspends a `webMethods` messaging trigger because all retry attempts have failed, Integration Server executes the resource monitoring service to determine if all the resources are available. The default is 60 seconds.

For more information about resource monitoring services, see the *Publish-Subscribe Developer's Guide*.

watt.server.trigger.preprocess.monitorDatabaseOnConnectionException

Indicates whether Integration Server schedules a system task to monitor the document history database when a `ConnectionException` causes a transient error occurs during trigger preprocessing. A `ConnectionException` indicates that the document history database is not enabled for exactly-once processing or is not properly configured.

When this property is set to `true`, Integration Server schedules a system task that executes an internal service that monitors the connection to the document history database. When the document history database is enabled for exactly-once processing or is properly configured, the internal service indicates that the connection to the document history database is available.

Then, Integration Server resumes the trigger and re-executes it when the connection to the document history database is available.

When this property is set to false, Integration Server does not schedule a system task to check for the database's availability and will not resume the trigger automatically. You must manually resume the trigger after configuring the document history database properly.

The default is false

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.trigger.preprocess.suspendAndRetryOnError

Indicates whether Integration Server suspends a trigger if an error occurs during the preprocessing phase of trigger execution. The preprocessing phase encompasses the time from when the trigger retrieves the document from its local queue to the time the trigger service executes. When this property is set to true, the trigger property **On Retry Failure** is set to **Suspend and retry later**, or the trigger property **On Transaction Rollback** is set to **Suspend and recover**, Integration Server suspends a trigger if one of the following occurs during preprocessing:

- A transient error occurs because the document history database is not available when Integration Server performs duplicate detection for the trigger. For more information about transient error handling during trigger preprocessing, see *webMethods Service Development Help*.

If the document history database is properly configured, Integration Server suspends the trigger and schedules a system task that executes a service that checks for the availability of the document history database. Integration Server resumes the trigger and re-executes it when the service indicates that the document history database is available.

If the document history database is not properly configured, Integration Server throws a `ConnectionException`. To determine how Integration Server handles a `ConnectionException` during trigger preprocessing, see `watt.server.trigger.preprocess.monitorDatabaseOnConnectionException`.

- The document resolver service ends because of an `ISRuntimeException`. Integration Server suspends the trigger and schedules a system task to execute the trigger's resource monitoring service (if one is specified). Integration Server resumes the trigger and retries trigger execution when the resource monitoring service indicates that the resources used by the trigger are available. If a resource monitoring service is not specified, you will need to resume the trigger manually (via the Integration Server Administrator or the `pub.trigger*` services).

When this property is set to false and the trigger property **On Retry Failure** is set to **Throw Exception** or the trigger property **On Transaction Rollback** is set to **Recover only**, Integration Server does not suspend the trigger if a trigger preprocessing error occurs. If the document history database is not available, Integration Server

- If the trigger specifies a document resolver service, Integration Server executes the document resolver service to determine the status of the document. If the document resolver service ends because of an `ISRuntimeException`, Integration Server assigns the document a status of In Doubt, acknowledges the document, and uses the audit subsystem to log the document.

- If a document resolver service is not specified, Integration Server assigns the document a status of In Doubt, acknowledges the document, and uses the audit subsystem to log the document.

Note: Integration Server uses the audit subsystem to log documents for webMethods messaging triggers.

The default value for `watt.server.trigger.preprocess.suspendAndRetryOnError` is `false`.

Important:

You must restart Integration Server for changes to this parameter to take effect.

For more information about building a resource monitoring service, see the *Publish-Subscribe Developer's Guide* or *Using webMethods Integration Server to Build a Client for JMS*. For more information about transient error handling during trigger preprocessing, see *webMethods Service Development Help*.

watt.server.trigger.removeSubscriptionOnReloadOrReinstall

Specifies whether Integration Server deletes document type subscriptions for webMethods messaging triggers when the package containing the trigger reloads or an update of the package is installed. If this property is set to `true` (the default) and a package reloads or an update of the package is installed, Integration Server deletes and then recreates any document type subscriptions for webMethods messaging triggers in the package. (If Integration Server connects to a Broker, Integration Server deletes and recreates the subscriptions on the trigger client on the Broker.) This creates a small window of time during which the document type subscriptions do not exist. During this window, the trigger will not receive documents to which it normally subscribes.

If this property is set to `false`, Integration Server does not delete and then recreate document type subscriptions for webMethods messaging triggers when the package reloads or is updated. Although Integration Server creates new document type subscriptions for triggers, Integration Server does not modify existing subscriptions. Specifically, if a trigger deleted a document type subscription, the subscription will not be removed when the package reloads or is updated. Consequently, when this property is set to `false`, the trigger might receive document types to which it no longer subscribes because the deleted document type subscriptions still exist on the trigger client on the Broker. When working with a 6.5.2 version of webMethods Broker, you can use My webMethods to delete the obsolete document type subscriptions from the trigger client on the Broker. The default is `true`.

Note:

This property does not affect webMethods messaging triggers running in a cluster of Integration Servers.

watt.server.trigger.reuseSession

Indicates whether instances of a webMethods messaging trigger use the same session on Integration Server when the document locale is the same as the default locale of Integration Server. When this property is set to `true`, Integration Server checks the locale of the document before processing it. If the document locale is the same as the default locale of Integration Server, or no locale is specified, the trigger uses a shared session. If the document locale is different from the default, then Integration Server creates a new session for the trigger to use to process that document. When this property is set to `false`, Integration Server uses a new session for each instance of a trigger. The default is `false`.

Reusing sessions for a webMethods messaging trigger might improve performance. However, this property does not work with all adapters.

Note:

If you use the Adapter for SAP set `watt.server.trigger.reuseSession` to `false` if a concurrent trigger has a trigger service that executes multi-step SAP transactions that make calls to the `pub.sap.client.lockSession` and `pub.sap.client.releaseSession` services. These services require dedicated sessions.

watt.server.trigger.suspendOnAuditErrorWhen

Specifies when Integration Server should suspend a webMethods messaging trigger. A trigger can be suspended when both the following occur:

- The trigger service fails.
- The trigger service cannot write audit data to the audit database because an audit exception occurs.

When Integration Server suspends a webMethods messaging trigger, it halts document processing and document retrieval for the trigger. Suspending the trigger prevents Integration Server from acknowledging the document and prevents the messaging provider from discarding the document. After suspending the trigger, Integration Server monitors the connection to the audit database and resumes the trigger (document processing and document retrieval) when the connection to the audit database is re-established. Integration Server retrieves the unacknowledged document from the messaging provider and attempts to process it again.

Set the `watt.server.trigger.suspendOnAuditErrorWhen` parameter to one of the following values:

Specify	To
Never	Indicate that Integration Server does <i>not</i> suspend a trigger if the trigger service ends because of an error and an audit exception occurs when the trigger service attempts to write data to the audit database.
Error	Indicate that Integration Server suspends a trigger if the trigger service ends because of an error and an audit exception occurs when the trigger service attempts to write data to the audit database.
ErrorPipeLineEnabled	Indicate that Integration Server suspends a trigger if the trigger service ends because of an error, the trigger service is configured to include the service pipeline in the audit log, and an audit exception occurs when the trigger service attempts to write data to the audit database. The default is <code>ErrorPipelineEnabled</code> .

Important:

You must restart Integration Server for changes to this parameter to take effect.

The `watt.server.trigger.suspendOnAuditErrorWhen` configuration parameter only applies when the audit subsystem is configured to write data synchronously.

watt.server.tspace.location

Specifies the absolute directory path of the hard disk drive space in which the Integration Server is to temporarily store large documents rather than keep them in memory. Each file that the

Integration Server stores in this directory is given the name DocResxxxxx.dat, where xxxxx is a value that can vary in length and character. Specify the absolute directory path to a directory on the same machine as the Integration Server. The default value is JVM's temporary directory (i.e., the value of java.io.tmpdir).

Example: If you want the Integration Server to use the LargeDocTemp directory on your D drive, specify the following:

```
watt.server.tspace.location=D:\LargeDocTemp
```

If you have a cluster of Integration Servers, each one must have its own Tspace.

Important:

You must restart Integration Server after you modify the value of this property.

watt.server.tspace.max

Specifies the maximum number of bytes that can be stored at any one time in the hard disk drive space that you defined using the `watt.server.tspace.location` property. If the Integration Server attempts to write a large document to the hard disk drive space that will cause the number of bytes you specify to be exceeded, an error message is displayed on the server console, and the document is not stored. Specify a positive whole number of bytes. The default value is 52,428,800 bytes (50 MB).

Example: To set the maximum number of bytes that can be stored to 30,000,000 bytes, specify the following:

```
watt.server.tspace.max=30000000
```

Tip:

The size of the hard disk drive space for temporarily saving documents will vary based on the number of documents that you process concurrently and the size of the documents that you process. For example, if your typical concurrent document load is 10, you would need a hard disk drive space that is 10 to 15 times the combined size of the documents being processed concurrently.

Important:

You must restart the Integration Server after you modify the value of this property.

watt.server.tspace.timeToLive

Specifies the amount of time in milliseconds that documents can remain in the Tspace before they are deleted. If a document remains in the Tspace longer than the value defined by this parameter, the document expires and becomes eligible for removal from the Tspace. Expired Tspace documents are removed when the Tspace begins to run low on free disk space. The default is 0.

Important:

You must restart the Integration Server after you modify the value of this property.

watt.server.txMail

Specifies the email address of an administrator to notify when guaranteed delivery capabilities are disabled due to an error (for example, if the Integration Server encounters a disk full condition or if the audit-trail log is full). There is no default.

watt.server.tx.cluster.jobPendingWait

Specifies the number of seconds that Integration Server waits for a guaranteed delivery job to complete when retrieving the results of a job with `TContext.invokeTx()` or `TContext.retrieveTx()`. The default is 60 seconds.

Important:

For changes to this parameter to take effect, you must reinitialize guaranteed delivery as described in [“Reinitializing Guaranteed Delivery” on page 745](#) or restart Integration Server.

watt.server.tx.cluster.lockBreakSecs

Specifies the number of seconds a cluster server waits before breaking a lock on a job in a cluster job store. The default is 120.

watt.server.tx.cluster.lockTimeoutMillis

Specifies the number of milliseconds a cluster server sleeps between attempts to place an update lock on a job in a cluster job store. The default is 100.

watt.server.tx.heuristicFailRetry

Specifies whether the Integration Server is to re-execute services for guaranteed delivery transactions in the job store that are pending when the Integration Server is restarted after a failure. If a transaction is pending, the service began execution before the Integration Server failed.

If the setting is `true`, the Integration Server resets the transaction status from **pending** to **new**, and the service will be re-executed. If the setting is `false`, the Integration Server resets the transaction status from **pending** to **fail** to indicate the heuristic failure, and the service will not be re-executed. The default is `true`.

watt.server.tx.sweepTime

Specifies the number of seconds between sweeps (clean up) of the job store for inbound guaranteed delivery transactions. The server sweeps the job store to remove expired transactions. The default is 60.

watt.server.um.producer.transaction.commitRetryCount

Specifies the number of attempts made by Integration Server in publishing a guaranteed document to a Universal Messaging server, after the initial attempt at publishing fails because of a transaction failure. The maximum number of retries is nine. If you try to assign a value greater than nine, Integration Server automatically sets the value of the property to nine. The default is zero.

When setting a retry value, you must ensure that the value of the total transaction timeout does not exceed the value of the `MaxTransactionTime` property. The total transaction timeout value is calculated by multiplying the total number of retry attempts with the value of the `EventTimeout` property. For example, if the retry value is set to 9, and the `EventTimeout` is set to 60s, the total transaction timeout is $60(9+1) = 600s$.

Note:

You can configure the values of the `MaxTransactionTime` and `EventTimeout` properties in the Universal Messaging Enterprise Manager.

watt.server.url.alias.partialMatching

Specifies whether Integration Server enables partial matching on URL aliases. If you set this server configuration parameter to `true` and define a URL alias in Integration Server Administrator, Integration Server enables partial matching on URL aliases. The default is `false`.

When partial matching is enabled Integration Server considers an alias a match if the entire alias matches all or part of the request URL, starting with the first character of the request URL path.

For information about partial matching and defining URL aliases, see [“Setting Up HTTP URL Aliases” on page 423](#). For specific information about partial matching for REST resources configured using the legacy approach, see *REST Developer’s Guide*.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.server.userFtpRootDir

Specifies the FTP root directory that the Integration Server will create at startup. When any Integration Server user logs into the FTP Listener, the server creates that user’s FTP home directory in this root directory, for example `FtpRoot/username`. You can specify *any* directory to be the root directory, including a mapped network directory. If this property is not defined, a default directory named `userFtpRoot` is created in your Integration Server home directory.

The user who connects to Integration Server through FTP listener is placed either in the default FTP root directory or the client user directory as defined in the `watt.server.login.userFtpDir` property.

Administrators, Replicators, and non-privileged users can perform put and get operations in the following directories:

This user	Can access
Administrator	<ul style="list-style-type: none"> ■ admin (in the Integration Server home directory) ■ The Administrator’s own user directory ■ The entire namespace for all packages, including <code>WmRoot</code> ■ All other user directories
Replicator	<ul style="list-style-type: none"> ■ The <i>Integration Server_directory</i> \instances\<i>instance_name</i>\replicate directory ■ Any services in the namespace that have Replicator ACL or lower ■ The Replicator’s own user directory
Non-privileged user	<ul style="list-style-type: none"> ■ The user’s own user directory ■ Any services that have an ACL at or below the level of the user’s ACL

When a user completes a put command in his or her own user directory (that is, when the STOR command is completed on the server side but before the server acknowledges the client with return code 226), an event is fired to notify interested parties by publishing a `pub.client.ftp:putCompletedNotification` document to the `webMethods Broker`. EDI packages will subscribe to this document and will retrieve the file just put onto the server.

Note:

The STOU command is not supported on the Integration Server. However, it is supported for clients. See the following built-in services in the *webMethods Integration Server Built-In Services Reference*: `pub.client.ftp`, `pub.client.ftp:put`, and `pub.client.ftp:mput`.

watt.server.ws.71xHandlerChainBehavior

Specifies whether Integration Server uses the 7.1x handler chain processing behavior when executing handlers for a web service descriptor created in 7.1x. When set to true, Integration Server uses the handler chain processing behavior available in Integration Server version 7.1x for a 7.1x web service descriptor. When set to false, Integration Server uses the handler chain processing behavior available in Integration Server 8.0 for a 7.1x web service descriptor. The default is false.

watt.server.ws.defaultNamespace

This is an internal property. Do not modify.

watt.server.ws.defaultPrefix

Specifies the prefix assigned to the namespace

`http://www.webMethods.com/2001/10/soap/encoding` for use in details for SOAP faults. The default is `webM`.

watt.server.ws.responseTNS.from.request

For a provider web service descriptor for which the **Pre-8.2 compatibility mode** property is set to true, specifies whether the target namespace of the top-level element in the response uses the target namespace of the top-level element in the request. When set to true, a web service response sent by the Integration Server uses the target namespace of the top-level element in the request as the top-level namespace of the response. While this is incorrect behavior, it provides backward compatibility. If existing web service clients failed because the client expected a response that used the namespace of the top-level element of the request and you want to allow existing clients to continue to process responses without any changes to the client, set `watt.server.ws.responseTNS.from.request` to true.

When set to false, Integration Server uses the target namespace of the provider web service descriptor as the namespace of the top-level element in the response. This is the correct behavior. The default value is false.

Note:

After upgrading to Integration Server 9.10 or later from a version of Integration Server earlier than 9.10, if you have web service providers that specify **Pre-8.2 compatibility mode** = true with existing web service clients that expect the namespace of the response to match the namespace of the request, they may fail due to the namespace mismatch. Software AG recommends regenerating those web service clients using a WSDL document retrieved from current provider web service descriptor. Alternatively, set `watt.server.ws.responseTNS.from.request` to true, which will revert to behavior of using the namespace of the request on the response rather than using the namespace as specified in the current WSDL document. While this behavior may be backward compatible, it may generate a response that does not match the current WSDL of the provider web service descriptor. For this reason, Software AG discourages setting `watt.server.ws.responseTNS.from.request` to true.

Note:

This parameter is deprecated because the web services implementation introduced in Integration Server 7.1 is deprecated as of Integration Server 10.4. The web services implementation introduced in Integration Server version 7.1 handles web service descriptors that run in pre-8.2 compatibility mode.

watt.server.ws.security.timestampMaximumSkew

Specifies the maximum number of seconds that the web services client and host clocks can differ so that the timestamp expiry validation does not fail. Integration Server validates the inbound SOAP message only if the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.

You can use this parameter to set the timestamp maximum skew value when Integration Server uses WS-Security policy to implement security and if you have not set a value in the **Timestamp Maximum Skew** field in the Create web Service Endpoints Alias screen. The value must be a positive integer or zero. The default is 300 seconds.

watt.server.ws.security.timestampPrecisionInMilliseconds

Specifies whether the timestamp placed in the Timestamp element of the security header of an outbound message is precise to seconds or milliseconds. If you set the precision to milliseconds, Integration Server uses the timestamp format `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'`. If you set the precision to seconds, Integration Server uses the timestamp format `yyyy-MM-dd'T'HH:mm:ss'Z'`.

You can use this parameter to set the precision when Integration Server uses WS-Policy to implement WS-Security and if you have not set a value in the **Timestamp Precision** field in the Create Web Service Endpoints Alias screen. Set this property to `true`, if you want the timestamp precision set to milliseconds. Set this property to `false`, if you want the timestamp precision set to seconds. The default is `true`.

watt.server.ws.security.timestampTimeToLive

Specifies the time-to-live value for an outbound message in seconds. Integration Server uses this value to set the expiry time in the Timestamp element of outbound messages.

You can use this parameter to set the time-to-live value when Integration Server uses WS-Policy to implement WS-Security and if you have not set a value in the **Timestamp Time to Live** field in the Create web Service Endpoints Alias screen. The value must be an integer greater than 0. The default is 300 seconds.

watt.server.ws.security.usernameTokenTTL

Specifies the permitted time difference, in seconds, between the time when the UsernameToken was created (as provided in the `wsu:Created` element) and the time when it reaches the server. You can use this parameter to configure the username token TTL at the global level. The default is 300 seconds.

You can also configure this property for individual endpoint aliases using the **WS Security Properties** section of the **Settings > Web Services** pages. The value set in the **Settings > Web Services** pages overrides the value in this configuration setting.

watt.server.ws.security.usernameTokenFutureTTL

Specifies whether the permitted time difference between `wsu:Created` elements with a timestamp that is in the future and the Integration Server clock. Integration Server considers such requests as valid if the time at which the request was created does not exceed the time at which it reaches Integration Server by the value (in seconds) given in this setting. The default is 60 seconds.

You can also configure this property for individual endpoint aliases using the **WS Security Properties** section of the **Settings > Web Services** pages. The value set in the **Settings > Web Services** pages overrides the value in this configuration setting.

watt.server.wsdl.debug

Specifies whether Integration Server prints debug information to standard out and stack traces to standard error while generating or consuming WSDL. When set to `true`, Integration Server prints debug information to standard out and stack traces to standard error. The default is `false`.

watt.server.xml.encoding

Specifies the encoding that Integration Server must use when processing incoming XML files. If an encoding is not defined in the XML header, Integration Server attempts to process the XML file using the charset encoding of the http or ftp request. If charset encoding is not available in the request header, then Integration Server uses the character encoding specified in the `watt.server.xml.encoding` server configuration parameter. There is no default value for this parameter.

Note:

If you have configured Integration Server to use the character encoding specified in the `watt.server.fileEncoding` parameter to process incoming XML files previously, ensure that the value of `watt.server.fileEncoding` parameter is set to the same value specified for `watt.server.xml.encoding`. If you have not configured `watt.server.fileEncoding` for processing XML files previously, after installing this fix or after upgrading to a higher version of Integration Server, you can configure `watt.server.xml.encoding` to process incoming XML files. You can use `watt.server.fileEncoding` to process all files other than incoming XML files.

Important:

You must restart Integration Server for changes to this parameter to take effect.

watt.server.xml.enforceEntityRef

Specifies whether the server will throw an exception when the XML parser detects a malformed entity. If the value is set to `true`, the server will throw an exception when it detects a malformed entity in an XML or DTD. If the value is set to `false` (the default), the server will allow malformed entities and does not throw an exception.

watt.server.xml.xmlNodeToDocument.keepDuplicates

Specifies whether the `pub.xml:xmlNodeToDocument` service keeps additional occurrences of an element in an XML node when arrays are not created (the `makeArrays` input parameter is set to `false`). This parameter also determines whether or not the `pub.event.eda:eventToDocument` service keeps additional occurrences of an element in the XML document passed into the service. When set to `true`, the document produced by the `pub.xml:xmlNodeToDocument` service contains multiple occurrences of the element. When set to `false`, the document produced by the `pub.xml:xmlNodeToDocument` service keeps only the last occurrence of the element. The default is `true`.

For example, suppose the following XML node is provided as input to the `pub.xml:xmlNodeToDocument` service:

```
<?xml version="1.0" encoding="UTF-8"?><myDoc><e1
e1Attr="attrValue1">e1Value1</e1><e2>e2Value</e2><e1
e1Attr="attrValue2">e1Value2</e1></myDoc>
```

When `watt.server.xml.xmlNodeToDocument.keepDuplicates` is set to `true`, the `pub.xml.xmlNodeToDocument` service produces this document:

The diagram shows an XML document tree. The root is 'document', which contains attributes '@version' (1.0) and '@encoding' (UTF-8). It contains a child element 'myDoc', which in turn contains two child elements 'e1'. The first 'e1' has an attribute '@e1Attr' with value 'attrValue1' and a child '*body' with value 'e1Value1'. The second 'e1' has an attribute '@e1Attr' with value 'attrValue2' and a child '*body' with value 'e1Value2'. Below the second 'e1' is another element 'e2' with value 'e2Value'.

document	
@version	1.0
@encoding	UTF-8
myDoc	
e1	
@e1Attr	attrValue1
*body	e1Value1
e1	
@e1Attr	attrValue2
*body	e1Value2
e2	e2Value

When `watt.server.xml.xmlNodeToDocument.keepDuplicates` is set to `false`, the `pub.xml.xmlNodeToDocument` service produces this document:

The diagram shows an XML document tree where the first 'e1' element from the previous diagram has been replaced by the second one. The root 'document' has the same attributes. 'myDoc' now contains only one 'e1' element, which has the attribute '@e1Attr' with value 'attrValue2' and the child '*body' with value 'e1Value2'. The 'e2' element remains at the bottom with value 'e2Value'.

document	
@version	1.0
@encoding	UTF-8
myDoc	
e1	
@e1Attr	attrValue2
*body	e1Value2
e2	e2Value

Note that only the last `<e1>` element in the source XML is retained in the resulting document.

watt.server.xml.xmlNodeToDocument.makeArrayforWS

Specifies how Integration Server decodes duplicate elements contained in an `anyType` element. Set `watt.server.xml.xmlNodeToDocument.makeArrayforWS` to `true` if you want Integration Server to create an array for duplicate elements contained in an element of type `anyType`. Set this parameter to `false` if you want Integration Server to leave duplicate elements as separate, repeated elements in the element defined to be of type `anyType`. When set to `false`, Integration Server does not create an array for elements that appear more than once in the element defined to be of type `anyType`. The default is `false`.

watt.ssl.

watt.ssl.accelerator.provider

Enables the use of the SSL accelerator provided with a T1/T2 processor on a Solaris 10 OS machine. The only value you can specify with this parameter is `SunPKCS11-Solaris`. To use this accelerator, Integration Server must be running with JVM 1.5 and the HSM Based Keystore field must be set to `true` on the **Security > Keystore > Create Keystore Alias** screen.

If you do not specify this parameter, SSL ports will be able to use the `nCipher` accelerator only. To use this accelerator, the HSM Based Keystore field must be set to `True` on the **Security > Keystore > Create Keystore Alias** screen.

watt.ssl.entrust.toolkit.ssl.fragmentblockcipher

Specifies whether the Entrust library is to fragment SSL records when a block cipher is used. The Entrust library included with Integration Server addresses an SSL vulnerability identified in the US-CERT Vulnerability Note VU#864643 (<http://www.kb.cert.org/vuls/id/864643>). The Entrust library changes the way SSL records are fragmented when using a block cipher such as AES. When a block cipher is used, the Entrust library breaks up the SSL record into two records: a 1-byte record and a record consisting of the remaining bytes. This change prevents the exploit from working.

If you need to disable fragmentation of the SSL records for interoperability, you can disable the fragmentation feature by setting the `watt.ssl.entrust.toolkit.ssl.fragmentblockcipher` parameter to `false`. The default value of this property is `true`.

watt.ssl.iaik.debug

Indicates whether Integration Server should log SSL handshake communication messages between the SSL client and SSL server in the server console. If set to `true`, Integration Server logs SSL handshake communication messages to the server console. The default is `false`.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.ssh.

watt.ssh.jsch.ciphers

Specifies a list of ciphers that Integration Server supports for communication with an SFTP server. When establishing a connection to an SFTP server, these are the ciphers that Integration Server, acting as an SFTP client, presents to the SFTP server.

The default values are: `aes256-ctr,aes192-ctr,arcfour,arcfour128,arcfour256`.

To include any other ciphers, append the cipher to the comma-separated list. To remove a cipher, delete the cipher from the list. You may want to remove a cipher if the cipher has known vulnerabilities.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.ssh.jsch.kex

Specifies a comma-separated list of the key exchange algorithms that will appear in the **Preferred Key Exchange Algorithms** list when creating or editing an SFTP server alias. Specify only the key exchange algorithms that are not deemed to be vulnerable. Do not list key exchange algorithms deemed to be vulnerable. The **Preferred Key Exchange Algorithms** list will display the key exchange algorithms in the same order in which the configuration parameter value lists the algorithms. The default is `""`, which means that Integration Server displays all key exchange algorithms in the **Preferred Key Exchange Algorithm** list.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.ssh.jsch.logging

Enables JSch logging. When this property is set to `true`, all the logs from JSch will be printed on Integration Server console. The default is `false`.

watt.ssh.jsch.mac_c2s

Specifies the order of message authentication code (MAC) algorithms for client to server transmission. The default value is the default order of MAC algorithms that Integration Server acting as an SFTP Client supports. To disable use of a MAC algorithm for client to server transmission, delete it from the `watt.ssh.jsch.mac_c2s` value.

Note:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

watt.ssh.jsch.mac_s2c

Specifies the order of message authentication code (MAC) algorithms for server to client transmission. The default value is the default order of MAC algorithms that Integration Server acting as an SFTP Client supports. To disable use of a MAC algorithm for server to client transmission, delete it from the `watt.ssh.jsch.mac_s2c` value.

Note:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Note:

- All `watt.ssh.jsch.*` parameters, except `watt.ssh.jsch.logging`, are deprecated. Do not use the deprecated parameters because the preferred key exchange algorithms, ciphers, and MAC algorithms are configured from the user interface.
- Integration Server uses the `watt.ssh.jsch.logging` server configuration property to enable logging for both versions of the SFTP client.

watt.tx.

watt.tx.defaultTTLMinutes

Specifies the default time-to-live (TTL) value for outbound guaranteed delivery transactions. Specify the number of minutes you want the server to maintain outbound transactions in the job store when a service initiating an outbound transaction does not specify a TTL value. The default is 30.

watt.tx.disabled

Specifies whether you want to disable the use of guaranteed delivery for outbound transactions. By default, the server allows the use of guaranteed delivery for outbound transactions. The default is `false`.

watt.tx.heuristicFailRetry

Specifies whether Integration Server is to re-execute services for guaranteed delivery transactions in the job store that are pending when Integration Server is restarted after a heuristic failure. If the transaction status is pending, it means that the service began execution before Integration Server failed.

If `watt.tx.heuristicFailRetry` setting is `true`, Integration Server resets the transaction status from pending to new, and Integration Server will retry the service. When the setting is `true`, a request to execute a service can only fail if the transaction expires before Integration Server executes the service.

If the setting is `false`, Integration Server resets the transaction status from pending to fail to indicate the heuristic failure, and Integration Server does not retry the service. When the setting is `false`, a request to execute a service can fail due to a heuristic failure or due to the transaction expiring.

The default is `true`.

watt.tx.jobThreads

Specifies the number of client threads you want to make available in a thread pool to service pending requests in the outbound guaranteed delivery job store. The default is 5.

watt.tx.retryBackoffTime

Specifies the number of seconds to wait after a service request failure before the Job Manager resubmits the request to execute the service to the Integration Server. The default is 60.

watt.tx.sweepTime

Specifies the number of seconds between sweeps of the job store of outbound guaranteed delivery transactions. The server sweeps the job store to identify transactions that it needs to submit. The default is 60.

watt.tx.vm.id

Specifies an Integration Server ID. Use this parameter when multiple Integration Servers are running on the same host machine and they all use guaranteed delivery. By specifying a unique ID for each Integration Server, you prevent the creation of duplicate guaranteed delivery transaction IDs. The value must be an integer from 0 through 2147483647. The default is 0.

watt.um

watt.um.clientLog.level

Determines the information that is written to the Universal Messaging client log file, called `umClient.log`. Each level outputs log entries with that level or higher. Valid values are `trace`, `debug`, `info`, `warn`, `error`, `fatal`, and `off`. The default is `error`.

The `watt.um.clientLog.level` parameter displays the value specified for the **Log Level** parameter in the **Settings > Logging > Edit UM Client Logger Details** screen in Integration Server Administrator. Software AG recommends that you use the **Log Level** parameter in Integration Server Administrator to set the logging level for the Universal Messaging client log file instead of the `watt.um.clientLog.level` server configuration parameter.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.um.clientLog.size

Maximum size of the Universal Messaging client log file measured in megabytes (MB). When this size is reached, Integration Server rolls the file over to a backup called `umClient.log.number` and creates a new file. The default is 10.

The `watt.um.clientLog.size` parameter displays the value specified for the **Log Size** parameter in the **Settings > Logging > Edit UM Client Logger Details** screen in Integration Server Administrator. Software AG recommends that you use the **Log Size** parameter in Integration Server Administrator to set the logging level for the Universal Messaging client log file instead of the `watt.um.clientLog.size` server configuration parameter.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.um.clientLog.fileDepth

Number of backup log files to keep on disk when using log rolling for the Universal Messaging client log file (see the `watt.um.clientLog.size` property). When this number is reached, the oldest file is deleted. The default is 2.

The `watt.um.clientLog.fileDepth` parameter displays the value specified for the **Log Depth** parameter in the **Settings > Logging > Edit UM Client Logger Details** screen in Integration Server Administrator. Software AG recommends that you use the **Log Depth** parameter in Integration Server Administrator to set the logging level for the Universal Messaging client log file instead of the `watt.um.clientLog.fileDepth` server configuration parameter.

Important:

If you change the value of this property, you must restart Integration Server for the change to take effect.

watt.wm.tnextdc.

watt.wm.tnextdc.configVersion

This is an internal parameter. Do not modify.

watt.wmcloud.

watt.wmcloud.hybridConnectivityAlert.mail

Specifies the list of email addresses to which Integration Server sends emails about errors and warnings related to the connection status between Integration Server and webMethods Cloud. Use a semicolon (;) as a delimiter to specify multiple email addresses. By default, the parameter value is blank.

Note:

This server configuration parameter is introduced for PIE-81173 in IS_10.5_Core_Fix23.

watt.wmcloud.hybridConnectivityAlert.notifications

Specifies whether Integration Server generates notifications and notifies users when a tenant connection or an account for webMethods Cloud is disconnected or restored. The parameter can have one of the following values:

Value	Description
alert	Integration Server displays a notification in Integration Server Administrator.
email	Integration Server sends an email to the user.
none	Integration Server does not notify the user.

Note:

If you are using any value other than alert, email, or none, Integration Server notifies the user through a notification in Integration Server Administrator, followed by an email. By default, the parameter value is alert.

This server configuration parameter is introduced for PIE-81173 in IS_10.5_Core_Fix23.

watt.wmcloud.listeners.monitoringInterval

Specifies the interval, in milliseconds, at which Integration Server monitors on-premise listeners to ensure that each enabled webMethods Cloud account has an active listener on the on-premise Integration Server. If the monitoring thread finds a listener that is not running, the monitoring thread attempts to start the listener. The `watt.wmcloud.listeners.monitoringInterval` server configuration parameter determines the interval at which the monitoring thread executes. The default value of the parameter is 300000 milliseconds (5 minutes).

watt.wmcloud.listeners.maxIdleTime

Determines the length of time, in milliseconds, that an on-premise listener for a webMethods Cloud account can be idle before it gets recreated. The default value of this parameter is 1800000 milliseconds (30 minutes).

C Environment Variables for Use with Docker

■ Environment Variables	1184
-------------------------------	------

Environment Variables

Integration Server and Microservices Runtime use environment variables (ENV variable) to support performing tasks while Integration Server or Microservices Runtime runs in a Docker container. Environment variables are included in the `docker run` command immediately before the image name. You can also use these environment variables when starting Integration Server or Microservices Runtime from the command line.

Note:

Unless otherwise specified, an environment variable for use with Microservices Runtime can be used with an Integration Server equipped with an Microservices Runtime license.

The following table identifies the environment variables supported with Integration Server or Microservices Runtime.

Environment Variable	Description
EXTERNALIZE_PACKAGES	<p>When set to true, instructs the Integration Server running in the Docker container to load the packages located in one of the following directories at startup:</p> <ul style="list-style-type: none"> ■ <code><HOST_DIR>/<SERVICE_NAME>/packages</code> ■ <code><HOST_DIR>/<INSTANCE_NAME>/packages</code> <p>Where <code>HOST_DIR</code>, <code>SERVICE_NAME</code>, and <code>INSTANCE_NAME</code> are the values set for the ENV variables of the same name.</p> <p>If <code>SERVICE_NAME</code> is supplied, Integration Server looks in <code><HOST_DIR>/<SERVICE_NAME>/packages</code>. Only if <code>SERVICE_NAME</code> is not supplied, does Integration Server look in <code><HOST_DIR>/<INSTANCE_NAME>/packages</code>.</p> <p>The default value of <code>EXTERNALIZE_PACKAGES</code> is false.</p> <p>The content of the packages directory must be a folder where the folder name is the package name. The contents of the <code>packageName</code> folder must match the structure of Integration Server packages. That is the packages located in the packages directory cannot be an archive file, such as <code>*.zip</code> or <code>*.7z</code>. For information about the structure of an Integration Server package, see “How the Server Stores Package Information” on page 657.</p>
HOST_DIR	<p>The path to the mounted directory on the HOST machine to which to write the files. Passing the <code>HOST_DIR</code> parameter and value or setting it as an ENV variable with the <code>docker run</code> command externalizes the logs and configuration artifacts of Integration Server.</p>

Environment Variable	Description
JAVA_MAX_MEM	<p>The maximum heap size. The default value is 1024 MB. Use the <code>JAVA_MAX_MEM</code> environment variable to override the value set for Microservices Runtime in <i>Integration Server_directory</i> /bin/setenv.bat sh file without having to edit the file itself.</p> <p>Note: The <code>JAVA_MAX_MEM</code> environment variable is for use with Microservices Runtime only.</p>
JAVA_MIN_MEM	<p>The minimum heap size. The default value is 256 MB. Use the <code>JAVA_MIN_MEM</code> environment variable to override the value set for Microservices Runtime in <i>Integration Server_directory</i> /bin/setenv.bat sh file without having to edit the file itself.</p> <p>Note: The <code>JAVA_MIN_MEM</code> environment variable is for use with Microservices Runtime only.</p>
PERSIST_LOGS	<p>If set to true, the Integration Server running inside the Docker container persists the log files to <i>HOST_DIR/SERVICE_NAME/logs</i> where <i>SERVICE_NAME</i> is replaced by <i>INSTANCE_NAME</i> if the <code>SERVICE_NAME</code> environment variable is not specified. Integration Server externalizes the logs located in the <i>Integration Server_directory</i> /instances/<instanceName>/logs and profiles/IS_<instanceName>/logs directories. If set to false, the Integration Server running inside the Docker container does not externalize the log files. The default is true.</p>
PERSIST_CONFIGS	<p>If set to true, the Integration Server running inside the Docker container persists the configuration files to the <i>HOST_DIR/SERVICE_NAME/config</i> directory where <i>SERVICE_NAME</i> is replaced by <i>INSTANCE_NAME</i> if the <code>SERVICE_NAME</code> environment variable is not specified. If set to false, the Integration Server running inside the Docker container does not externalize the config files. The default is true.</p>
SAG_IS_AUDIT_STDOUT_LOGGERS	<p>Specifies the audit loggers to write to the console (STDOUT). When an audit logger writes to STDOUT it is considered an auxiliary output which means the logger will still write to the specified log destination of file or database. Set to one of the following:</p>

Environment Variable	Description
	<ul style="list-style-type: none"> <li data-bbox="570 260 1292 359">■ A comma-separated list of the audit loggers that you want to write to STDOUT. For example: WMSESSION,WMERROR <p data-bbox="615 386 1292 485">The audit logger name is the first portion of the audit log file name. For a list of audit log files names, see <i>webMethods Audit Logging Guide</i> .</p> <li data-bbox="570 516 1292 653">■ A comma-separated list of the logger names as they appear in the Settings > Logging page in Integration Server Administrator. For example: Session Logger,Error Logger <li data-bbox="570 684 1292 747">■ ALL to write all log entries for all file-based audit loggers to STDOUT. <li data-bbox="570 779 1292 877">■ NONE to indicate that none of the file-based audit loggers will write to STDOUT. This is the default value. <p data-bbox="570 905 1073 930">The variable content is case-insensitive.</p> <p data-bbox="570 961 1292 1060">When writing audit log entries to STDOUT, Integration Server includes the logger name in the entry to identify the source of the entry.</p> <p data-bbox="570 1092 1292 1228">For information about using the SAG_IS_AUDIT_STDOUT_LOGGERS environment variable with Integration Server running in Docker container, see “Writing Audit Logs to the Console” on page 965.</p> <p data-bbox="570 1255 1292 1423">You can configure the delimiters used for fields and records in the audit log entries written to STDOUT using the <code>watt.server.audit.stdout.fieldDelimiter</code> and <code>watt.server.audit.stdout.recordDelimiter</code> server configuration parameters, respectively.</p>
SAG_IS_CONFIG_PROPERTIES	Specifies the location and name of the configuration variables template for use with Microservices Runtime or an Integration Server with licensed Microservices Runtime functionality.
SAG_IS_CONFIG_VARIABLES_DEBUG	When set to true, instructs Microservices Runtime to include debug level messages in the log generated during configuration variable initialization. When set to false, configuration log messages of severity Information and higher are included in the configuration variables log. The default is false.

Environment Variable	Description
	<p>For more information about configuration variables logging, see <i>Developing Microservices with webMethods Microservices Runtime</i> .</p>
SAG_IS_LICENSE_FILE	<p>Supplies a license file to use with Integration Server in place of the licenseKey.xml located in the <i>Integration Server_directory/instances/instanceName/config</i>. The specified license key must be a valid license file and in a location accessible by Integration Server. For an Integration Server running in a Docker container, the provided license key must be available on a mounted directory on the host file system. At start up, if a file exists at the location specified by SAG_IS_LICENSE_FILE, Integration Server copies the file from the specified location to <i>Integration Server_directory/instances/instanceName/config</i> and saves the file as licenseKey.xml</p> <p>Supplying the license file at start up can be useful when a Docker image for an Integration Server contains an expired license file or you want it specify an alternative license file for use with an existing Docker image. By using SAG_IS_LICENSE_FILE to supply the license file, you can change the license file without recreating a Docker image.</p>
SERVICE_NAME	<p>Name of a unique directory under HOST_DIR for persisting Integration Server artifacts. When supplied, the logs and config directories are created under the <i>HOST_DIR/SERVICE_NAME</i> directory and all artifacts of logs and config directories are persisted in the respective directories. If SERVICE_NAME is not supplied, Integration Server instance name will be used for the name of the unique directory that gets created under the <i>HOST_DIR</i> directory. Software AG recommends specifying a unique directory for each Docker container.</p>

D FIPS 140-2 Compliance

■ FIPS 140-2 Compliance	1190
-------------------------------	------

FIPS 140-2 Compliance

webMethods Integration Server Version 9.0 and later embeds the Entrust Authority Security Toolkit for Java 8, which has obtained FIPS 140-2 validation. FIPS (Federal Information Processing Standards) provides standards for information processing for use within the Federal government. The policy for Version 8 is available at the following:

<http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2012.htm#1839>

Many government and financial organizations require that their software be FIPS 140-2 compliant, which follows the current standards and guidelines for cryptographic information processing.

Note: Integration Server itself is *not* considered to be FIPS 140 certified.

Running Integration Server in FIPS 140-2-compliant mode ensures that it only uses FIPS compliant algorithms in the FIPS compliant modes. You can enable FIPS mode by setting the following extended setting on the Integration Server:

```
watt.security.fips.mode=true
```

Refer to “[Server Configuration Parameters](#)” on page 1017 for a detailed description of this server configuration parameter. Also, refer to “[Working with Extended Configuration Settings](#)” on page 127 for instructions on viewing and updating extended settings for the Integration Server.

In addition to running the server in FIPS compliant mode, you must follow the other instructions in the Entrust Cryptographic Module Security Policy. The instructions include implementing safeguards such as not allowing multiple users to access the computer and ensuring that the computer is physically protected. In particular, see section 5.4 of that document (“Operational Environment”). Depending on your organization's policies, you might also be required to use the same hardware, operating system, and JDK as was used in the Entrust approval.

FIPS mode encryption is only applicable to HTTPS or FTPS communications and S/MIME encryption/signing.

E Using NTLM Authentication when Integration Server Acts as the Client

■ Overview	1192
■ Using Java-Based NTLM Support	1192
■ Using Native NTLM Support via Integrated Windows Authentication	1193

Overview

When Integration Server executes services that access web pages, Integration Server behaves like a web client making a request to a web server. Integration Server supports NTLM (Windows NT LAN Manager) authentication on the connection from Integration Server to web servers that support NTLM authentication. When properly configured for NTLM client support, Integration Server responds to an NTLM challenge from a web server with the appropriate authentication credentials.

You can use Integration Server to enable NTLM authentication support for service requests where Integration Server acts as the client. Integration Server provides two variants of NTLM client support:

- Java-based NTLM for UNIX platforms and Windows systems.
- Native NTLM for Windows systems.

For Native NTLM, Integration Server uses Integrated Windows authentication as a means of authenticating its identity while establishing connections between Integration Server and web servers on an intranet.

This appendix explains how to use NTLM authentication with Integration Server when Integration Server acts as the client only.

Using Java-Based NTLM Support

By default, Integration Server uses Java-based NTLM support. The Java-based NTLM support can be used on UNIX as well as Windows platforms.

Integration Server responds to an NTLM challenge from a web server with the appropriate authentication credentials, whether Integration Server runs on Windows, UNIX, or another supported platform.

Java-based NTLM authentication in Integration Server has the following limitations:

- Java-based NTLM authentication supports NTLMv2 only.
- Java-based NTLM authentication does not support NTLMv1.
- Java-based NTLM authentication can be used with HTTP and HTTPS.
- Java-based NTLM authentication in Integration Server does not work with NTLM proxy servers.

When using Java-based NTLM client authentication, keep the following information in mind:

- You must provide the authentication credentials explicitly. While providing the authorization information, you must prefix the domain name followed by a backslash (\) before the username. For example, when using NTLM as the authentication type for an invocation of the `pub.client:http` service, you must specify a value for the user input parameter using the format:
domain_name\user_name

For more information about setting *auth* type in the `pub.client:http` service, see *webMethods Integration Server Built-In Services Reference*.

- The NTLM server must be configured to send the “NTLM” header and the “Negotiate” header. If Integration Server receives only the “Negotiate” header, the NTLM handshake will not take place.

Note:

If you intend to use Java-based NTML authentication, make sure native NTLM is disabled. By default, Integration Server uses Java-based NTLM authentication and native NTLM is disabled. For information about disabling native NTLM support, see [“Deactivating Integrated Windows Authentication” on page 1194](#).

Using Native NTLM Support via Integrated Windows Authentication

If Integration Server runs on Windows, Integration Server can use the “native” NTLM support which uses *Integrated Windows authentication* as a means of authenticating Integration Server’s identity. Integrated Windows authentication authenticates a user without requiring the transmission of actual passwords or sensitive account information across the network.

Which credentials Integration Server uses when responding to an Integrated Windows authentication request depends on whether Integration Server runs as a standalone application or as an NT service:

- If Integration Server runs as a standalone application, it uses the credentials of the logged in Windows user.
- If Integration Server runs as an NT service, it uses the local system rights for authentication. If you log on as a user, Integration Server uses the credentials associated with that session when responding to an Integrated Windows authentication

Native NTLM authentication in Integration Server has the following limitations:

- Native NTLM authentication is for Windows systems only.
- Native NTML authentication supports HTTP only.
- Native NTLM authentication is not supported for NTLM proxy servers.
- For Integration Server to use native NTLM support the web server must support Integrated Windows authentication. Microsoft Internet Information Server (IIS) is an example of a web server that supports Integrated Windows authentication.

To use native NTML authentication on Integration Server, you must first activate Integrated Windows authentication on Integration Server. For information about activating Integrated Windows authentication, see [“Activating Integrated Windows Authentication” on page 1194](#).

Activating Integrated Windows Authentication

Once Integrated Windows authentication is activated, Integration Server automatically responds to its requests.

> To activate Integrated Windows authentication

1. Open Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. If the WmWin32 package is not already enabled, enable it by clicking **No** in the **Enabled** column for this package.
4. In the list of packages, click **WmWin32**.

Note:

The WmWin32 package is deprecated as of Integration Server 7.1.

5. Click **Browse Services in WmWin32**.
6. In the list of services, click **wm.ntlm:reg**.
7. Click **Test reg**. The server displays the test screen for the win32.ntlm.reg service.
8. Click **Test (without inputs)**. The server activates Integrated Windows authentication.

Note:

If you want Integrated Windows authentication available whenever Integration Server is running, make the win32.ntlm:reg service a startup service for the Win32 package. For more information about assigning startup services, see *webMethods Service Development Help*.

Deactivating Integrated Windows Authentication

> To deactivate Integrated Windows authentication

1. Open the Integration Server Administrator if it is not already open.
2. In the **Packages** menu of the Navigation panel, click **Management**.
3. In the list of packages, click **WmWin32**.
4. Click **Browse services in WmWin32**.

5. In the list of services, click **wm.ntlm:unreg**.
6. Click **Test unreg**. The server displays the test screen for the win32.ntlm.unreg service.
7. Click **Test (without inputs)**. The server deactivates Integrated Windows authentication.

F Removing User Data from Integration Server

■ Removing User Data	1198
■ Removing References to a User Account	1198
■ Removing References to Email Addresses	1201
■ Removing Personal Data from Log Files	1202

Removing User Data

Data protection laws and regulations, such as the GDPR (General Data Protection Regulation) might require specific handling of user data, even after a user profile is removed. Additionally, employees or other clients with user accounts on Integration Server may request that any user identifying information such as user name, email addresses, or client IP addresses be removed from Integration Server. To comply with data protection requirements and user requests, in addition to deleting the user account, you may need to complete activities such as changing certificate mappings, revising server configuration parameter values, assigning new execution users to services, and editing log files.

Removing References to a User Account

Prior to deleting a user account you need to update all of the locations in Integration Server that reference the user account, including execution users, certificate mappings, and outbound connection configurations. If you delete the user account before updating functionality that depends on the user account to execute successfully, Integration Server may experience failures.

Software AG recommends that you complete user-related updates in this order:

1. Change client certificate mappings. An imported client certificate or CA (Certificate Authority) signing certificate is mapped to a user account. Before you delete a user account that is mapped to a client certificate, do *one* of the following:

- Change the certificate mapping. For more information see, [“Changing a Certificate Mapping” on page 521](#).
- Remove the certificate mappings stored in the database table IS_CERTIFICATE_MAP of ISInternal JDBC Pool. Because a user can be mapped to more than one certificate, you might need to delete multiple mappings. You can use the following database DELETE statement to quickly remove any user certificate mappings:

```
DELETE FROM IS_CERTIFICATE_MAP WHERE CERT_USER = “username”
```

Where *username* is the user account that you intend to delete.

2. Change the user account assigned to execute tasks, services, and triggers. You might need to update one or more of the following:

Update this asset	Specifically	Using
Scheduled tasks,	Run As User value on the Server > Scheduler > User Tasks > Modify Tasks page	Integration Server Administrator.
JMS triggers	Execution user property for the trigger	Designer

Update this asset	Specifically	Using
webMethods messaging triggers that receive messages from Universal Messaging	Execution user property for the trigger	Designer
webMethods messaging triggers that receive messages from Broker or messages published locally	Run Trigger Service As User field available on the Settings > Resources > Store Settings page	Integration Server Administrator
Enterprise Gateway rule that uses a service as a custom filter.	Run As User field for the custom filter on the Security > Enterprise Gateway > Rules > Rules > rulename > Edit page	Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server
Enterprise Gateway alerts that invoke a flow service to alert you of a rules violation.	Run As User field for the default alert options on the Security > Enterprise Gateway Rules > Edit Default Alert Options page	Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server
Email ports	Run services as user field on the Security > Ports > Edit Email Client Configuration page	Integration Server Administrator
File polling ports	Run services as user field on the Security > Ports > Edit File Polling Configuration page	Integration Server Administrator
Package subscriptions on the current server	Remote User Name field on the Packages > Publishing > Edit Subscriber page	Integration Server Administrator
Package subscriptions on a remote server	Local User Name field on the Packages > Subscribing > Edit Subscription page	Integration Server Administrator
WmCloud account settings	Run As User field on the webMethods Cloud > Accounts > Edit Account page	Integration Server Administrator

Note:

The above list does not include execution users assigned for adapters.

3. Change server configuration parameters that specify a user as the value, including:

- `watt.server.cache.prefetchUser`
- `watt.server.event.routing.runAsUser`
- `watt.server.eventHandlerUser`

Use the **Extended Settings** page in Integration Server Administrator to edit the server configuration parameters

Note:

The above list of server configuration parameters is not exhaustive and may not include parameters added via fixes or by a layered product such as an adapter.

4. Change outbound connection configurations. Any location in Integration Server, including any services, in which a user name is specified to establish an outbound connection. This may include the following:

- Remote server alias, which is editable via Integration Server Administrator.
- Web service endpoint alias, which is editable via the **Settings > Web Services > *endpointAliasName* > Edit** page in Integration Server Administrator.
- Messaging connection aliases, including:
 - JMS connection alias, which is editable via the **Settings > Messaging > JMS Connection Alias > Edit** page of Integration Server Administrator.
 - `webMethods` messaging connection alias, which is editable via the **Settings > webMethods Messaging Settings > Connection Alias > Universal Messaging Connection Alias > Edit** page of Integration Server Administrator
- Outbound HTTP calls such as those using `pub.client:http`, `pub.client:soapClient`, or web service connectors

5. Change configuration variables templates. A configuration variables template used with a Microservices Runtime image running in a Docker container or an on-premises Microservices Runtime might specify a user name for one of the key-value pairs. Using a text editor, edit the template to change the value of property key that specifies the user name you want to remove from Microservices Runtime. Property keys for a user name typically include the word “user” or “principal”.

Note:

If a Docker image for an Microservices Runtime includes `application.properties` template, then each Docker container created from the image contains the template too. The template is removed when the Docker container gets destroyed. If the containers do not get recycled periodically, you can attach to the container file system and scrub the user names from the `application.properties` template.

Note:

The configuration variables template feature is included in Microservices Runtime by default. An Integration Server equipped with an Microservices Runtime license can use the configuration variables feature as well.

6. Delete the user from Integration Server.
7. Repeat steps 2–7 for each Integration Server on which the user name exists. For example, if you use a cluster of Integration Servers, you need to repeat the steps for all servers in the cluster.

Note:

Client certificate mappings are stored in a database which is shared by a cluster. You do not need to repeat step 1 for every Integration Server in the cluster.

8. Recreate any Docker images for the affected Integration Server instances.

Removing References to Email Addresses

As part of removing references to a user from Integration Server, you may need to remove references to their email address as well. Review the following and remove or replace the email address as needed:

- Email addresses for notifications about critical information, new package releases, rules violations, and so forth. You might need to update email addresses for one or more of the following:

Update this asset	Specifically	Using
Email Notification settings	Internal Email field and the Service Email field on the Settings > Resources > Edit Resource Settings page	Integration Server Administrator
Password expiration notification email addresses.	Expiration Notice Email Addresses field on the Security > User Management > Password Expiration Settings > Edit page.	Integration Server Administrator
Enterprise Gateway alerts that send an email to alert you of a rules violation	Email Addresses field on the Security > Enterprise Gateway Rules > Edit Default Alert Options	Integration Server Administrator on the Integration Server acting as the Enterprise Gateway Server
Notification email addresses for package subscriptions on this Integration Server	Notification E-mail field on the Packages > Publishing > Edit Subscriber page	Integration Server Administrator

Update this asset	Specifically	Using
Notification email addresses for package subscriptions on a remote Integration Server	Notification E-mail and Automatic Pull E-mail on the Packages > Subscribing > Edit Subscription page	Integration Server Administrator

- Server configuration parameters that specify an email address as the value:
 - watt.server.email.from
 - watt.server.errorMail
 - watt.server.serviceMail
 - watt.server.txMail

Note:

The above list of server configuration parameters is not exhaustive and may not include parameters added via fixes or by a layered product such as an adapter.

Use the **Extended Settings** page in Integration Server Administrator to edit the server configuration parameters

- Services that invoke the `pub.client:smtp` service, `pub.replicator.notifyPackageReleaseservice`, or any other service that uses email addresses as input parameters.

Note:

After you remove or update email addresses used in Integration Server, you need to recreate any Docker images for the affected Integration Server instances.

Removing Personal Data from Log Files

Upon request, you may need to remove identifying personally identifiable information (PII) for a user from Integration Server log files. This data may include user name, email address, and client IP address. The following sections provide information about the PII in each log file, how to locate the data, and how to delete it.

Note:

Remove PII data from log files only after you have updated Integration Server to remove references to the user account and email address and deleted the user account.

Removing Personally Identifiable Information from the Server Log

The Integration Server server log contains information about operations and errors that occur on Integration Server. The amount of data that Integration Server and other layered products write to the server log depends on the logging levels in use.

At a minimum, Integration Server rotates the server log daily, but the server log can be configured to rotate based on size, resulting in multiple logs for a single day. When Integration Server rotates the server.log file, Integration Server renames the current log to use the archive file name and starts a new server.log file. The archive file name uses the format `server.log_yyyyMMdd_HHmmssSSSZ`, where `yyyyMMdd_HHmmssSSSZ` is the date and time the log file was created. Integration Server stores the server log and the server log archive files in the same directory.

The default name and location of the server log is: `Software AG_directory \ Integration Server_directory \ instances \ instance_name \ logs \ server.log`

The following table identifies the type of user data that might be written to the server.log, how to find it, and how to remove it.

Data	How to find and remove
User name	Use a text editor to perform a search and replace for the user name in the server.log or the archive files. For example, you could search the server.log files for the user name and replace the user name with an anonymous string or a blank string.
Client IP Address	<p>Integration Server rarely includes the client IP address in a server log message. To locate these messages, use a text editor to search the server log files for messages with the following message IDs and then remove or replace the client IP address in the messages.</p> <p>ISS.0053.0002C</p> <p>ISS.0053.0012C</p> <p>ISS.0053.0017C</p> <p>ISS.0138.0505E</p> <p>ISC.0037.0013D</p> <p>ISC.0064.0014T</p> <p>ISC.0064.0015T</p> <p>ISC.0064.0020T</p> <p>ISC.0064.0021T</p> <p>ISC.0064.0029T</p>
Email address	<p>Integration Server rarely includes email addresses in server.log messages. Use a text editor to perform a search and replace for the email address in the server.log or the archive files</p> <p>Integration Server logs the email address of the administrator for the message with the following message ID: ISC.0063.0070D</p>

Note:

When running a Docker image of Integration Server in a Docker container, Integration Server writes the server log to the console as well as to the server.log file. You may need to edit log files generated by the logging driver used with Docker. For example, the json-file logging driver, which is the default logging driver for Docker, captures everything written to STDOUT and writes it to a JSON file. The JSON log file may be rotated. When the Docker container is destroyed, the JSON log files are removed. As a result, you might not need to remove identifying user data from the JSON log files. If, however, you used Docker volumes to persist configuration and log files to a mounted directory on the host file system, you may need to clean the externalized log files as described above.

Removing Personally Identifiable Information from the Session Log

In the session log, the session audit record includes the user name and client IP address of the user making the connection to Integration Server. Where that data is located and how to remove it depends on whether the session log destination is a file or a database.

If the session log is a Use the following to locate and delete user data

File

File Location: *Software AG_directory \ Integration Server_directory \instances\instance_name\logs*

File Name: *WMSESSION_serverStartupTimestamp.log*

userid is 64-characters long. It is logged from column 528.

ip-address from which the user made the connection is 128 characters. It is logged from column 593.

Use a text editor to perform a search and replace for the userid and client IP address. For example, you could search the session log files for the user name and replace the user name with an anonymous string or a blank string.

Database

If the Session audit logger is configured to log to an external database identified by the ISCoreAudit functional alias, Integration Server writes the log message into the WMSESSION table of the ISCoreAudit database.

The USERID column contains the userid.

The SESSIONNAME column contains the ip-address from which the user made the connection.

Use the following SQL statements to remove the data:

```
DELETE FROM WMSESSION WHERE USERID = 'username'
```

Where *username* is the user account to delete from the log.

If the session log is a Use the following to locate and delete user data

```
DELETE FROM WMSESSION WHERE SESSIONNAME = 'ipAddress'
```

Where *ipAddress* is the client IP address to delete from the log.

Removing Personally Identifiable Information from the Service Log

In the service log, the service audit log record contains the user name of the user executing the service. Where that data is located and how to remove it depends on whether the service log destination is a file or a database.

If the service log is a Use the following to locate and delete user data

File

File Location: *Software AG_directory \ Integration Server_directory \instances\instance_name\logs*

File Name: *WMSERVICE_serverStartupTimestamp.log*

userid is 64-characters long. It is logged from column 1024.

Use a text editor to perform a search and replace for the userid. For example, you could search the service log files for the user name and replace the user name with an anonymous string or a blank string.

Database

If the Service audit logger is configured to log to an external database identified by the ISCoreAudit functional alias, Integration Server writes log message into WMSERVICE table of the ISCoreAudit database.

The USERID column contains the userid.

Use the following SQL statement to remove the data:

```
DELETE FROM WMSERVICE WHERE USERID = 'username'
```

Where *username* is the user account to delete from the log.

Removing Personally Identifiable Information from the Security Log

In the security log, the security audit log record includes the userid of the user performing the logged activity. Where that data is located and how to remove it depends on whether the security log destination is a file or a database.

If the security log is a **Use the following to locate and delete user data**

File File Location: *Software AG_directory \ Integration Server_directory \instances\instance_name\logs*

File Name: *WMSECURITY_serverStartupTimestamp.log*

userid is 64-characters long. It is logged from column 640.

Use a text editor to perform a search and replace for the userid. For example, you could search the security log files for the user name and replace the user name with an anonymous string or a blank string.

Database If the Security audit logger is configured to log to an external database identified by the ISCoreAudit functional alias, the log message are written into WMSECURITY table of the ISCoreAudit database.

The USERID column contains the userid.

Use the following SQL statement to remove the data:

```
DELETE FROM WMSECURITY WHERE USERID = 'username'
```

Removing Personally Identifiable Information from the SSL Session Log

In the SSL session log (*inboundSSLSessions.log*), the session information includes the user name and client IP address of the user who connects to Integration Server. SSL session log is always in a file format. Use the following information to locate the user data:

- File Location: *Software AG_directory \ Integration Server_directory \instances\instance_name\logs*
- File Name: *inboundSSLSessions.log*
- User id: userid is 64-characters long. It is logged from column 21.
- Client IP address: IP-address from which the user made the connection is 128 characters. It is logged from column 24.

To delete the user data, use a text editor to perform a search, and replace the userid and client IP address. For example, you could search the session log files for the user name and replace the user name either with an anonymous string or a blank string.

Removing Personally Identifiable Information from the Configuration Variables Log

The configuration variables log contains messages about the operations, warnings, and errors that occur while Microservices Runtime applies a configuration variables template at startup. This can include messages about setting a property key for a user name to a specific value. The configuration

variables log is overwritten at startup of an on-premise Microservices Runtime. The configuration variables log for a Microservices Runtime running in a Docker container is destroyed when the container is destroyed.

Microservices Runtime writes the configuration variables log messages to the console (STDOUT) and/or to the location: *Integration Server_directory* /instances/*instanceName*/logs/configurationvariables.log. A Microservices Runtime running in a Docker container always writes the configuration variables log messages to both locations.

To remove references to a user name or user ID from the configurationvariables.log, use a text editor to search the log for the user name or user ID. Then remove or replace the found data.

To remove references to a user name from the configurationvariables.log for a Docker container, you can attach to the container file system and then search for and remove or change the user name from the log.

If Microservices Runtime writes the configuration variables log to the console (STDOUT), then you need to remove or replace the user name information from the destination that captures the STDOUT stream.

- Docker captures the STDOUT stream in a file which, by default, is a JSON file. You may need to edit log files generated by the logging driver used with Docker. For example, the json-file logging driver, which is the default logging driver for Docker, captures everything written to STDOUT and writes it to a JSON file. The JSON log file may be rotated. When the Docker container is destroyed, the JSON log files are removed. As a result, you might not need to remove identifying user data from these JSON log files. If, however, you used Docker volumes to persist configuration and log files to a mounted directory on the host file system, you may need to clean the externalized log files as described above.
- For an on-premise Microservices Runtime, the STDOUT stream is written to *Software AG_directory* /profiles/*IS_instanceName*/logs/wrapper.log

Use a text editor to search the wrapper.log as well as the other log files in the logs folder for user information. Then, remove or replace the found data.

Removing Personally Identifiable Information Logged by Axis2, Kerberos, SAML, and other Third Party Libraries

Third party libraries used by Integration Server can log user data, particularly at debug or trace log levels. These messages go into the log files stored in *Software AG_directory* /profiles/*IS_instanceName*/logs folder. The wrapper.log located in this directory captures all the message that go to System.out. Use a text editor to search the wrapper.log as well as the other log files in the logs folder for user information (user names, client IP addresses, email addresses). Then, remove or replace the found data.

G Wireless Communication with the Integration Server

■ Overview	1210
■ How Does the Integration Server Communicate with Wireless Devices?	1210
■ Using URLs for Wireless Access to the Integration Server	1211

Overview

The webMethods Integration Server can receive requests from and send responses to Internet-enabled wireless devices. A wireless device requests information from the Integration Server the using a URL. The responses sent by the server contain WML (Wireless Markup Language) content or HDML (Handheld Device Markup Language) content. Examples of wireless devices that the Integration Server can communicate with include Internet-enabled wireless phones and Internet-enabled personal digital assistants.

You might want to use a wireless device to communicate with the Integration Server to:

- Check inventory levels at your company or at a supplier.
- Place an order or check the status of an existing order.
- Receive order confirmation for an order submitted with a wireless device.
- Send or receive notification to alert subscribers to trade fulfillments of security price changes.
- Collect statistics about your Integration Server by using event handlers that send information to wireless devices.
- Request an HDML or WML page stored on the Integration Server.

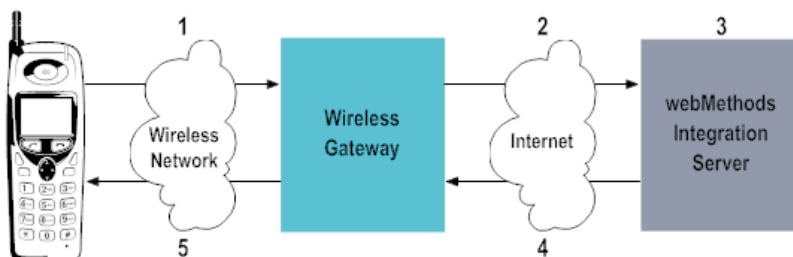
You access the Integration Server from a wireless device by entering a URL in the web browser of wireless device. The URL can invoke a service on the Integration Server or can request a WML or HDML page stored on the Integration Server.

How Does the Integration Server Communicate with Wireless Devices?

The Integration Server communicates with wireless devices by means of a wireless gateway. The wireless gateway (sometimes called a WAP gateway) converts a request from a wireless device to an HTTP request. The wireless gateway also converts the HTTP response from the Integration Server to a format understood by the web browser or micro-browser on the wireless device.

The following diagram illustrates how the Integration Server communicates with an Internet-enabled wireless device.

Communication Between the Integration Server and a Wireless Device



Stage	Description
1	A user requests a URL using a web browser on a wireless device such as a wireless phone or a personal digital assistant (PDA). The URL indicates the service to be invoked or identifies the requested WML or HDML page. The wireless device sends an encoded request to the wireless gateway.
2	The wireless gateway (such as a Phone.com's Up.Link Server or Nokia Active Server) decodes the request from the wireless device, creates an HTTP or HTTPS request (depending on what is specified in the URL) for the specified URL, and sends it to the Integration Server.
3	The Integration Server does one of the following depending on what the user requested in the URL: Executes the service specified in the URL and inserts the service results into the assigned WML or HDML output template. -OR- Retrieves the WML or HDML page requested in the URL.
4	The Integration Server sends an HTTP or HTTPS response to the wireless gateway.
5	The wireless gateway removes the HTTP or HTTPS header from the response and sends an encoded response containing the HDML or WML content to the wireless device. The web browser on the wireless device decodes the response and displays the WML or HDML results.

For more information about wireless gateways and wireless protocol, see www.wapforum.org.

Using URLs for Wireless Access to the Integration Server

To use a wireless device to access information or invoke services on the Integration Server, you need to use the device's web browser to enter or select a URL. The following sections explain how to invoke a service with a URL and how to request a WML or HDML page with a URL.

Note:

Some web browsers for wireless devices place limitations on the length of a URL that a user can enter or select. Make sure any WML or HDML pages that you create for use with wireless devices are compliant with browser requirements.

Note:

To minimize the amount of user input and therefore reduce the possibilities for input errors, embed hyperlinks to URLs in the WML or HDML page.

Invoking a Service with a URL

You can use a URL to invoke a service from an Internet-enabled wireless device. You can request a URL by entering the URL into the web browser directly or by selecting a link for the URL that

is embedded into a HTML or WML page. In either case, the URL needs to be in the following format:

1
2
3
4
5

```
http://localhost:5555/|invoke|folderName.subFolderName|serviceName?variable=value&variable=value
```

Item	Description
1	<p>Identifies the name and port number for the Integration Server on which the service you want to invoke resides.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Important: For wireless access, the server name (localhost) must be a registered domain name; that is, the server needs to be accessible via the Internet.</p> <p>Important: Many wireless gateways use port 80 as the default registered port number. If you want to use a different port number, make sure to register the server name and port number with the wireless gateway. (For security reasons, Software AG discourages using port numbers below 1024. For more information, see "Setting Up Aliases for Remote Integration Servers" on page 115.</p> </div>
2	Specifies the required keyword "invoke", which tells the Integration Server that the URL identifies a service that is to be invoked.
3	Identifies the folder in which the service to invoke resides. Separate subfolders with periods. This field is case sensitive. Be sure to use the same combination of upper and lower case letters as specified in the folder name on the Integration Server.
4	Identifies the service that you want to invoke. This field is case sensitive. Be sure to use the same combination of upper and lower case letters as specified in the service name on the Integration Server.
5	<p>Specifies the input values for the service. Specify a question mark (?) before the input values. The question mark signals the beginning of input values. Each input value is represented as a <i>variable=value</i> pair. The <i>variable</i> portion is case sensitive. Be sure to use the same combination of upper and lower case letters as specified in your service. If your service requires more than one input value, separate each <i>variable=value</i> pair with an ampersand (&).</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Only specify this part of the URL when using the HTTP GET method.</p> </div>

For more information about invoking a service with a URL, see "Building a Browser Based Client" in *webMethods Service Development Help*.

Note:

If you use the URL to invoke a service, make sure that the service applies the output to the appropriate template type (WML or HDML). For more information about creating output templates, see the *Dynamic Server Pages and Output Templates Developer's Guide*.

Requesting a WML or HDML Page with a URL

You can use an Internet-enabled wireless device to request a WML or HDML page stored on the Integration Server. By entering a URL in the web browser of a wireless device or by selecting a hyperlink to a URL, you can access any WML or HDML page stored in the following directory:

Integration Server_directory \instances\instance_name\packages\packageName\pub

Where *packageName* is the name of the package in which the WML or HDML file is saved.

The URL you enter in the web browser needs to adhere to the following format:

`http://localhost:5555/packageName/pub/filename`

Item	Description
1	<p>Identifies the name and port number for the Integration Server on which the file you want to request resides.</p> <p>Important: For wireless access, the server name (localhost) must be a registered domain name; that is, the server needs to be accessible outside via the Internet.</p> <p>Important: Many wireless gateways use port 80 as the default registered port number. If you want to use a different port number, make sure to register the server name and port number with the wireless gateway. (For security reasons, Software AG discourages using port numbers below 1024. For more information, see “Setting Up Aliases for Remote Integration Servers” on page 115.)</p>
2	Identifies the package in which the WML or HDML file you want to request resides.
3	<p>Specifies the pub directory. WML and HDML files that can be served to wireless devices need to reside in this directory.</p> <p>Note: You do not need to specify the pub directory. Integration Server automatically looks in pub for the requested file if no directory is specified.</p>
4	Identifies the file you want to request.

For example, the following URLs access the hello.wml file from the pub directory for the Wireless package:

<http://localhost:5555/Wireless/pub/hello.wml>

-OR-

<http://localhost:5555/Wireless/hello.wml>

H Masking Session IDs in Integration Server

Session and Server Logs

Integration Server allows authorised administrator users to secure Session IDs in the session and server logs. When you enable the option of securing Session IDs, Integration Server uses the asterisk (*) character to mask the Session ID strings in the logs. Therefore, no user accessing the log entries can view the actual Session IDs. This increases the security of Integration Server sessions by preventing malicious access to an active session of a genuine user.

Note:

The Session IDs are not masked when Integration Server writes the session log entries to the audit logs. For more information about setting up audit logging, see *webMethods Audit Logging Guide*.

The authorised administrator users can secure Session IDs by editing the `maskSessionID.properties` file in the Integration Server installation directory.

Important:

- Only those users who belong to the Administrators ACL and have write access to the `maskSessionID.properties` file can edit its contents. All the other administrator users will only have read access to the particular properties file.
- The authorized administrator user must manually edit the contents of the `maskSessionID.properties` file.

> To mask Session IDs

1. Navigate to the `maskSessionID.properties` file that resides in the *Integration Server_directory* \instances*instance_name*\config\security\session\ directory.
2. Open `maskSessionID.properties` and set `maskSessionID = true`.
3. Save your changes and restart Integration Server.

Note:

You must restart Integration Server whenever you update the contents of the `maskSessionID.properties` file.

I Server Log Facilities

■ About Server Log Facilities	1218
■ Integration Server	1218
■ WmJSONAPI Package	1224
■ WmXSLT Package	1224
■ Flat File	1224

About Server Log Facilities

Server logging provides operational and error information from Integration Server's major subsystems, called *facilities*.

You can select the facilities that you want Integration Server to include in the server log. For more information about the server log, see ["Setting Up the Server Log" on page 213](#).

Integration Server

The table below gives the list of Integration Server facilities for which the server logs information.

Facility	Information and errors related to	
0000	General Debugging	Types of information that do not belong to other listed facilities.
0001	License Manager	License manager of Integration Server.
0002	LDAP Connection	LDAP server connections.
0003	Database Connection Manager	Database connections.
0004	JDBC Connection	JDBC connection pools.
0006	Server SSL Interface	Integration Server SSL connection.
0007	Authorization	Authorization in Integration Server.
0008	NIS Connection	Network information services server that contains network resource information just like LDAP servers.
0009	Certificate	Handling, managing, and processing certificates.
0011	Proxies	Proxy servers.
0012	Authentication	Authentication to Integration Server.
0013	Content Handler	Content handlers.
0014	Server	Integration Server startup and shutdown, thread pooling, and other low-level parts of Integration Server. Includes license manager-related error managers that were not included in the 0001 License Manager facility.
0015	Service Manager	Flow and coded services that run in Integration Server.
0016	Service Caching	Service caching.
0017	Ehcache	Ehcache configuration.

Facility		Information and errors related to
0018	Utility classes	Utility classes used by Integration Server.
0019	OpenID Connect	OpenID Connect.
0020	Circuit Breaker	Circuit breaker configured for a service.
0021	Integration Cloud	Integration Cloud activities.
0022	Code Generation	Code generated using Designer for implementing a service or for calling a service from another service or client.
0024	User Manager	User management.
0025	Server Initialization	Server initialization.
0026	Services	Flow and coded (for example, Java) services that run in Integration Server.
0027	Server Configuration	HSM based keystores and nCiphers.
0028	Packages	Loading, unloading, and other operations of packages.
0030	Admin API	The Integration Server Administrator API.
0033	Cluster Manager	Clustered Integration Servers.
0036	Client Context	Client transactions.
0037	Guaranteed Delivery Context	Guaranteed delivery transactions.
0038	HTTP Header	HTTP headers.
0039	HTTP Request	HTTP requests to Integration Server.
0040	HTTP Response	HTTP responses from Integration Server.
0041	HTTP Cookie	HTTP cookies.
0042	XML Parser	XML parser.
0043	XML Parse Stream	XML parser reading XML data from a stream.
0046	HTTP Listener	HTTP listeners.
0047	HTTPS Listener	HTTPS listeners.
0048	Misc. Server Error Messages	Miscellaneous errors that were not included in the 0014 Server facility.
0049	Flow Operation	Flow operations.

Facility		Information and errors related to
0050	Flow Map	MAP steps in flow services and mapping of pipeline data in INVOKE steps.
0051	Port Manager	Integration Server ports.
0053	HTTP Dispatch	Processing of HTTP requests.
0054	HTTP Document Handler	Document handlers registered for an HTTP response.
0055	Java Services	Java services.
0056	Mailer	Mailer that Integration Server uses.
0057	Deployer	Deployer.
0059	Service Thread	Service thread.
0061	Remote Servers	Remote servers.
0062	Core Services	XML and XQL processing.
0063	Transaction Job Manager	Transaction manager.
0064	Network Services	Network services.
0068	Email Listener	E-mail listeners.
0070	Listeners	Listeners defined on Integration Server.
0071	FTP Listener	FTP listeners.
0072	Reporter	DSP processing.
0075	WmDB	WmDB package.
0076	Coders	Coders such as IDataBinCoder that Integration Server uses to serialize Java objects into bytes for storing into files or for sending on the network, and vice versa.
0077	com.wm.util	com.wm.util coder.
0079	Enterprise Gateway	webMethods Enterprise Gateway Server activity.
0080	Asynchronous Connection API	Network sockets.
0081	Namespace	Namespace configurations.
0082	XML Schema	XML Schema.
0085	WmRoot Package	WmRoot package.
0087	WmWin32 Package	WmWin32 package.

Facility	Information and errors related to
0088	SOAP SOAP messages.
0090	pub Flow Services Certificates processing, pub.flow services, and WSDL processing.
0091	WmRoot Admin Services WmRoot admin services.
0094	Repository V4 Repository Version 4.
0095	Audit Log Manager Audit log manager.
0096	JDBC Connection Manager JDBC connection pools.
0097	Broker Document Type Synchronizer Synchronizing publishable document types with the associated Broker document type on the Broker.
0098	Dispatcher Triggers, JMS clients on Broker, and Broker clients.
0099	Broker Transport Layer Broker connectivity.
0100	Web Container Tomcat JSP engine and the component that hosts the Tomcat JSP engine.
0101	Process Runtime WmPRT package.
0105	Cross-referencing and Latching Cross-referencing and latching in Integration Servers that belong to a cluster.
0106	Join Manager Joins used in trigger processing.
0109	Server Statistics Integration Server statistics for session and thread activity.
0114	Adapter Runtime Adapter runtime facilities.
0115	Adapter Runtime (Listener) Adapter listeners that use adapter connections to connect to adapter resources.
0116	Adapter Runtime (Notification) Adapter notifications including polling and listener notifications.
0117	Adapter Runtime (Adapter Service) Adapter services that define operations that the adapter will perform on adapter resources.
0118	Adapter Runtime (Connection) Adapter connections that contain parameters that adapter notifications and listeners use to connect to an adapter resource.
0119	Monitor webMethods Monitor.
0120	Monitor (Database Layer) webMethods Monitor database layer.

Facility	Information and errors related to
0121	Adapter Runtime (SCC Transaction Manager) Adapter Runtime (JCA System Contract Component Transaction Manager).
0123	Basis FSData File system used as an underlying storage system for other Integration Server components such as the repository, temporary store, and flat file logging. Provides detailed information about errors encountered by those components.
0125	Supervisory Control Mechanism Resources in Integration Server to prevent resource (memory, thread, or disk) starvation. Currently only controls resources assigned to publish/subscribe and the thread pools.
0126	Adapter Runtime (SCC Connection Manager) Adapter Runtime (JCA System Contract Component Connection Manager).
0127	Basis Transient Store Temporary stores, such as the one used for logging.
0128	Exactly Once Processor Exactly-once processing for webMethods messaging triggers or JMS triggers.
0129	.NET Package .NET services.
0130	Mainframe Package Mainframe package.
0131	Diagnostic Port Integration Server diagnostic port.
0132	Version Control System Version Control System Integration feature in Integration Server.
0133	DOM Implementation Used when XML Node is created as DOM Node or used as a DOM Node
0134	JMS Subsystem JMS Subsystem.
0135	JNDI Client Configuration JNDI client configuration, including JNDI provider management and connectivity.
0137	User Task Scheduler User created scheduled tasks.
0138	Asset Publisher Asset Publisher package.
0139	Keystore Keystores.
0140	Designer Software AG Designer
0141	Web Service stack Web service stack.
0142	Thread Kill Thread kill facility that you can use to cancel or kill service threads in cases where a service is unresponsive.

Facility	Information and errors related to	
0144	XML Security Services	The pub.security.xml services.
0147	SFTP Client	Integration Server functioning as an SFTP client.
0149	Enterprise Gateway Rules	webMethods Enterprise Gateway rules.
0150	SPM	Software AG Platform Manager subsystem in Integration Server.
0151	Monitoring (CMP)	Content monitoring platform.
0152	CORS	CORS (Cross-Origin Resource Sharing) support.
0153	Dispatcher (WM Messaging)	Triggers that receive messages or documents from Universal Messaging.
0154	Protocol Buffer Encoding (Universal Messaging)	Protocol buffer encoding and decoding.
0155	Hot Deployment	Hot deployment of Integration Server assets.
0156	Event Routing	The pub.event.routing services in WmPublic.
0157	Assets	The pub.assets services in WmPublic.
0158	OData	OData services.
0159	Digital Event Services	Creating, sending, and receiving events with Digital Event Services.
0160	Search and Refactor	Search and refactor of Integration Server assets.
0161	WebSocket	WebSocket endpoint provisioning and callback invocations.
0162	Ehcache IData	Ehcache is used as an underlying storage system for other Integration Server components such as the repository, temporary store, and flat file logging. Provides detailed information about errors encountered by those components.
0163	JSON Web Token	JSON Web Token authentication and configuration.
0164	JSON Validation	Validation of JSON content against a schema.
0165	WebSocket Listener	WebSocket listener provisioning and life cycle.
0166	REST	REST resources and REST API descriptors.
0167	HTTP Interceptor	The initialization and operation of the HTTP Interceptor framework

Facility		Information and errors related to
0168	Messaging (Enhanced Logging)	Sending and receiving of messages when enhanced logging is configured.
0169	Account Locking	User account locking.
0170	JSON Schema	JSON schema.
0171	Automatic deployment of packages	Installing, activating and other operations of automatic package deployment.
0172	GraphQL	GraphQL operations.
0173	Common Messaging	MQTT messaging.

WmJSONAPI Package

The table below gives the list of WmJSONAPI package facilities for which the server logs information.

Facility		Information and errors related to
0801	CORE	Information and errors related to webMethods JSONAPI support.

WmXSLT Package

The table below gives the list of WmXSLT package facilities for which the server logs information.

Facility		Information and errors related to
0001	SAX	SAX parsing-related information.
0002	JAXP	JAXP-related information, including messages for XML parsers and XSLT engines.
0003	Services	XSLT services public service information.
0004	Admin Services	XSLT services non-public administrative service information.

Flat File

The table below gives the list of flat file facilities for which the server logs information.

Facility		Information and errors related to
0000	FlatFile Logger	Flat file processing.
0001	FlatFile Listener	Flat file polling listeners.
0002	FlatFile Parser	Flat file parsers.

