

webMethods Deployer User's Guide

Version 9.9

October 2015

This document applies to webMethods Product Suite Version 9.9 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 © 2004-2015 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 <http://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 <http://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 <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Table of Contents

About this Guide	11
Document Conventions.....	11
Online Information.....	12
Concepts	13
About webMethods Deployer.....	14
Runtime-Based Deployment.....	14
Overview of Runtime-Based Deployment.....	14
Repository-Based Deployment.....	15
Composites.....	16
Build Script.....	16
Overview of Repository-Based Deployment.....	16
Creating Projects.....	17
Deployment Sets.....	18
Unresolved Dependencies.....	18
Target Servers.....	18
Deletion Sets.....	19
Mapping Projects.....	19
Deploying Projects.....	20
Checkpoint and Roll Back.....	20
Transactional Deployment.....	20
Concurrent and Sequential Deployment.....	21
Deployer Interfaces.....	21
Automating Project Creation.....	22
Logging.....	23
Getting Started	25
Getting Started with Runtime-Based Deployment.....	26
Getting Started with Repository-Based Deployment.....	27
Building Composites for Repository-Based Deployment	29
Overview.....	30
Before Building Composites.....	30
Adding Assets to the Source Directory.....	31
Installing the Asset Build Environment.....	31
Setting the Properties for the Build.....	32
Setting Build Properties.....	32
Setting VCS Checkout Properties.....	39
Changing JVM Memory Settings.....	40
Running the Build Script and Rebuilding the Index.....	41
Running the Build Script.....	45
Rebuilding the Index.....	46

Preparing BPM Assets for Repository-Based Deployment.....	46
Differences Between Manual Process Generation and Deployed Processes.....	47
About the Deployment of Generation Receipts.....	47
About the Redeployment of Processes.....	47
About the Deployment of Process Images.....	48
Preparing the BPM Process Development Asset Deployment Environment.....	48
About build.xml Files.....	48
Configuring the Process Project build.xml File.....	49
Running the Build Script in a Process Project Directory.....	51
Next Steps.....	51
Starting Deployer and Connecting to Servers.....	53
Starting Deployer.....	54
Connecting to webMethods Servers.....	54
Connecting to Integration Servers and Trading Networks Servers.....	56
Connecting to Optimize Servers.....	57
Connecting to Application Platform Servers.....	58
Connecting to My webMethods Servers.....	59
Connecting to Event Servers.....	62
Connect to Broker Servers.....	63
Connect to BPM Process Model Servers.....	65
Connect to Universal Messaging Servers.....	67
Connecting to EDA Deployment Endpoints.....	69
Connecting to Business Rules Integration Servers.....	70
Connecting to Business Rules My webMethods Servers.....	71
Connecting to a Repository for Repository-Based Deployment.....	72
Editing Properties for Source and Target Servers.....	72
Creating Target Groups.....	73
Next Steps.....	76
Creating and Managing Projects.....	77
Enabling or Disabling Deployer GUI Audit Logging.....	78
Setting Default Properties for All Projects.....	79
Setting the Dependency Checking Default.....	79
Setting the Project Locking Default.....	79
Setting General Deployment Defaults.....	79
Setting the Defaults for Integration Server and Trading Networks Projects.....	82
Creating a Project.....	85
Exporting and Importing Project Properties.....	89
Permissions for Performing Tasks in Projects.....	90
Adding and Viewing Instructions or Notes About a Project.....	92
Editing Settings for an Individual Project.....	93
Deleting a Project.....	93
Next Steps.....	93
Defining a Deployment Set.....	95

Creating a Deployment Set.....	96
Identify Source Servers for the Deployment Set.....	97
Identifying Source Servers for Runtime-Based Deployment.....	97
Identifying Source Repository for Repository-Based Deployment.....	98
Next Steps.....	99
Adding Assets for Runtime-Based Deployment.....	101
Before Adding Assets for Runtime-Based Deployment.....	102
Adding Assets to Broker, ProcessModel, MWS, or Optimize Deployment Sets.....	102
About MWS Deployment Sets.....	102
About ProcessModel Deployment Sets.....	103
Adding Assets to a Broker, ProcessModel, MWS, or Optimize Deployment Set.....	103
Adding Assets to an IS & TN Deployment Set.....	104
Deploying ACLs.....	104
Deploying ACLs Associated with My webMethods Server Groups.....	104
Deploying ACLs Associated with LDAP Groups.....	105
Adding Integration Server Administration Assets.....	105
Adding Integration Server Packages.....	105
Adding an Entire Package.....	106
Adding Package Components.....	106
Adding Package Files.....	107
Setting Package Properties.....	108
Adding webMethods Files.....	111
Adding Trading Networks Assets.....	112
Excluding Common Assets.....	113
Resolving Dependencies.....	113
Manually Adding Dependencies to a Package Component in an IS & TN Deployment Set...	115
Removing Process Models from a Deployment Set.....	116
Next Steps.....	116
Adding Assets for Repository-Based Deployment.....	117
Overview.....	118
Selecting Composites.....	118
Selecting Individual Assets from Composites.....	119
Resolving Dependencies.....	120
Resolving Dependencies Automatically.....	120
Resolving Dependencies Manually.....	121
Resolving Conflicts.....	123
Deploying ACLs.....	124
Deploying ACLs Associated with My webMethods Server Groups.....	124
Deploying ACLs Associated with LDAP Groups.....	124
Next Steps.....	124
Defining a Deletion Set.....	127
About Deletion Sets.....	128
Creating a Deletion Set.....	128

Identifying Servers.....	130
Identifying Servers for a Runtime-Based Deletion Set.....	130
Identifying Servers for a Repository-Based Deletion Set.....	131
Adding Assets to a Deletion Set.....	131
Adding Assets to a Runtime-Based Deletion Set.....	131
Adding Assets to a Repository-Based Deletion Set.....	132
Adding Packages to Deletion Sets.....	132
Resolving Dependencies in Repository-Based Deletion Sets.....	134
Exporting and Importing Deletion Set Definitions.....	134
Next Steps.....	135
Building a Runtime-Based Deployment Project.....	137
Creating a Build.....	138
Rebuilding a Build.....	139
Exporting and Importing a Build.....	139
Next Steps.....	140
Mapping a Project.....	141
About Mapping a Project.....	142
Mapping a Project to Target Servers and Target Groups.....	142
Exporting and Importing a Map.....	145
Substituting Configuration Values.....	146
Substituting Configuration Values by Asset.....	147
Substituting Configuration Values by Target Server (Runtime-Based).....	147
Substituting Configuration Values by Target Server (Repository-Based).....	148
Exporting and Importing Substitute Configuration Values.....	148
Deploying a Project.....	151
Overview.....	152
Preparing Integration Server to Stream Large Repository-Based Projects.....	152
Generating a Checkpoint.....	153
Generating an Automatic Checkpoint.....	153
Generating a Checkpoint Manually.....	153
Deploying a Project.....	154
Post-Deployment Tasks.....	157
Rolling Back Target Servers.....	157
Rolling Back Target Servers Automatically.....	158
Rolling Back Target Servers Manually.....	158
Using Deployer Commands.....	161
Overview.....	162
Installing Command Line Interface Only.....	162
Creating and Running Scripts.....	162
Specifying Log On Parameters.....	165
Creating a Configuration File for Log On Parameters.....	167
Error Handling and Logging.....	167

General and Project Commands.....	168
About.....	168
Deleting a Project.....	168
Displaying Project Properties.....	168
Exporting Deletion Sets from a Project.....	169
Importing Deletion Set Definitions into a Project.....	169
Exporting Project Properties.....	170
Importing Project Properties.....	170
Help.....	171
Listing Builds, Maps, or Deployment Candidates for a Project.....	171
Locking Projects.....	172
Unlocking Projects.....	172
Build Commands.....	172
Creating a Project Build.....	172
Listing Builds for a Project.....	173
Displaying Contents of a Build.....	173
Displaying Substitute Configuration Values for Integration Server Assets in a Build.....	173
Displaying Contents of a Build File.....	174
Displaying Substitute Configuration Values for Integration Server Assets in a Build File.....	174
Exporting a Build from a Project.....	175
Importing a Build File into a Project.....	175
Listing Build Reports.....	176
Displaying a Build Report.....	176
Commands for Repository-Based Deployment.....	177
Rebuilding the Index with the Build Script.....	177
Map Commands.....	178
Listing All Deployment Maps.....	178
Exporting a Deployment Map from a Project.....	178
Editing a Deployment Map, Project Properties, or Substitute Configuration Values.....	179
Importing a Deployment Map Into a Project.....	179
Exporting Substitute Configuration Values for Integration Server Assets from a Deployment Map.....	180
Importing Substitute Configuration Variables for Integration Server Assets into a Deployment Map.....	181
Deleting a Deployment Map from a Project.....	182
Deployment Commands.....	182
Creating a Deployment Candidate.....	182
Displaying Information About a Deployment Candidate.....	183
Deleting a Deployment Candidate.....	183
Generating a Checkpoint.....	183
Simulating a Deployment.....	184
Deploying.....	184
Rolling Back Target Servers.....	185
Listing Simulation, Rollback, and Deployment Reports.....	186

Displaying a Simulation, Rollback, or Deployment Report.....	186
Automating Project Creation.....	187
Overview.....	188
Exporting Projects for Use in Project Automator.....	188
Using Handles Instead of Passwords.....	189
Creating Password Handles.....	189
Modifying Password Handle Associations.....	190
Deleting Password Handles.....	190
Error Handling and Logging.....	191
Root Tag.....	191
Identifying Deployer.....	192
Setting Up Aliases for Source and Target Servers.....	192
Setting Up Aliases for Source Repositories.....	193
Setting Up Aliases for Target Application Platform Deployment Endpoints.....	194
Setting Up Aliases for Source and Target webMethods Brokers.....	195
Setting Up Aliases for Source and Target Process Model Servers.....	200
Setting Up Aliases for Source and Target Integration Servers.....	202
Setting Up Aliases for Source and Target My webMethods Servers.....	204
Setting Up Aliases for Source and Target Optimize Servers.....	206
Setting Up Aliases for Target Event Servers.....	208
Setting Up Aliases for Target EDA Deployment Endpoints.....	209
Setting Up Aliases for Target Business Rules Integration Servers.....	211
Setting Up Aliases for Target Business Rules My webMethods Servers.....	213
Setting Up Aliases for Target Universal Messaging Servers.....	214
Setting Up Aliases for Target Groups.....	218
Creating Projects.....	220
Defining Deployment and Deletion Sets for Runtime-Based Deployment.....	221
Defining a Deployment Set for Repository-Based Deployment.....	224
Defining a Deletion Set for Repository-Based Deployment.....	228
Building a Project for Runtime-Based Deployment.....	229
Mapping a Project.....	229
Mapping a Runtime-Based Project.....	230
Mapping a Repository-Based Project.....	231
Creating a Deployment Candidate for Runtime-Based Deployment.....	233
Creating a Deployment Candidate for Repository-Based Deployment.....	233
Running Project Automator.....	234
Deploying Process Models with E-Forms.....	235
Deploying Process Models with E-Forms.....	236
Deploying Optimize Assets.....	239
Overview.....	240
Disabling Automatic Execution of DDL Statements.....	240
Deploying Optimize Assets in Static DB Schema Mode.....	241
Optimize Deployment Usage Notes.....	242

Potential Problems.....	243
Deployer Batch Size.....	243
Removing Assets from a Deployment Set.....	243
Two or More Deployment Sets for the Same Analytic Engine Using One Deployment Map.....	243
Executing DDL Statements for Two or More Analytic Engines.....	244
Deploying to Clustered Integration Servers.....	245
Overview.....	246
Setting Up Connections to Integration Servers in the Cluster.....	246
Creating the Target Group.....	247
Deployable Assets.....	249
Application Platform Assets.....	250
BPM Process Development Assets.....	252
Broker Assets.....	253
Business Rules Assets.....	254
EDA Assets.....	255
Event Server Assets.....	256
Integration Server Assets.....	257
Integration Server Administrative Assets.....	258
Adding Administrative Assets to the Source Directory.....	258
Global Values for Integration Server Administrative Assets.....	263
Integration Server Administrative Assets and Substitution Values.....	266
ACLs.....	266
Broker Settings.....	266
Cache Manager.....	268
Certificate Settings.....	269
Client Certificates.....	269
CSRF Guard Configuration.....	270
webMethods Enterprise Gateway Configuration.....	270
Extended Settings.....	271
File Access Control Configuration.....	271
Global Variables.....	272
Groups.....	272
JDBC Driver Alias.....	272
JDBC Pool Alias Configuration.....	273
JDBC Functional Alias.....	273
JMS Connection Alias.....	273
JNDI Alias.....	275
Kerberos Settings.....	275
Keystore Alias.....	276
LDAP Configuration.....	277
Metadata.....	278
Enhanced Parser.....	278
Ports.....	279

Proxy server alias.....	283
Proxy Server Bypass.....	284
Reliable Messaging Configuration.....	284
Remote Server Alias.....	287
SAML Token Issuer.....	288
Scheduled Tasks.....	288
SFTP Server Alias.....	289
SFTP User Alias.....	290
Truststore Alias.....	291
Users.....	292
Integration Cloud Accounts.....	292
Integration Cloud Applications.....	292
Integration Cloud Settings.....	293
Web Service Endpoint Alias.....	293
Web Service Policy.....	296
Integration Server Administrative Asset Dependencies.....	296
Integration Server Package Assets.....	297
About Integration Server Packages.....	297
Adding Package Assets to the Source Directory.....	297
Global Values for Integration Server Package Assets and Composites.....	299
Individual Values for Integration Server Package Assets.....	302
Adapter Runtime and .NET Service Assets.....	304
Adding Adapter Runtime and .NET Service Assets to the Source Directory.....	304
Adapter Runtime Assets.....	304
Adapter Runtime Asset Dependencies.....	306
.NET Asset.....	307
Mobile Support Assets.....	308
My webMethods Server Assets.....	308
Optimize Assets.....	313
Trading Networks Assets.....	314
Universal Messaging Assets.....	317
Other Assets.....	320

About this Guide

This guide explains how to use webMethods Deployer to deploy assets from source webMethods servers or development environments to target webMethods servers.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention <i>folder.subfolder:service</i> .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
<i>Italic</i>	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
Monospace font	Identifies text you must type or messages displayed by the system.
{ }	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

Software AG Documentation Website

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

Software AG Empower Product Support Website

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](#).

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.

1 Concepts

■ About webMethods Deployer	14
■ Runtime-Based Deployment	14
■ Repository-Based Deployment	15
■ Creating Projects	17
■ Mapping Projects	19
■ Deploying Projects	20
■ Deployer Interfaces	21
■ Automating Project Creation	22
■ Logging	23

About webMethods Deployer

webMethods Deployer is a tool you use to deploy user-created assets that reside on source webMethods runtimes or repositories to target webMethods runtime components (*runtimes*). For example, you might want to deploy assets you have developed on servers in a development environment (the source) to servers in a test or production environment (the target).

Note: You can deploy *user-created* assets using Deployer. You cannot deploy webMethods components that have been installed by the Software AG Installer as part of a product. For example, you can deploy Integration Server packages that have been created by users, but you cannot deploy Integration Server packages that were installed, such as WmPRT. If you want such components to reside on target runtimes, you must install them using the Software AG Installer.

Deployer supports two scenarios for deploying assets:

- In a *runtime-based deployment* scenario, Deployer deploys assets from webMethods runtimes to which Deployer is connected.
- In a *repository-based deployment* scenario, Deployer deploys assets built from sources in a development environment or VCS and stored on a repository.

Runtime-Based Deployment

In runtime-based deployment, you deploy assets directly from webMethods source runtimes to target runtimes.

You can deploy assets from and to these webMethods runtimes:

- webMethods Brokers.
- BPM (ProcessModel) runtimes. A ProcessModel runtime is an Integration Server that hosts the webMethods Process Engine and executes business processes.
- Integration Servers, including hosts for Trading Networks Servers.
- My webMethods Server.
- Optimize runtimes. An Optimize runtime is an Optimize Analytic Engine.

Overview of Runtime-Based Deployment

The runtime-based deployment process involves the following basic stages:

Stage 1 Define a deployment project.

You use Deployer to create connections to target and source webMethods runtimes. You then define a deployment set, for which you select user-created assets to include in the project directly from source webMethods runtimes.

For more information about projects and deployment sets, see ["Creating Projects" on page 17](#).

Stage 2 Build the project.

You build your project for deployment to the target runtimes. When you build a project, Deployer creates a file that contains the actual assets referenced in the project. If you later change the project, or if the build contains assets that you know have changed on the source runtimes, you can re-create the build to bring it up to date.

Stage 3 Map the contents of the project build to target runtimes.

For more information, see ["Mapping Projects" on page 19](#).

Stage 4 Deploy the assets in the project build to the target runtimes.

For more information, see ["Deploying Projects" on page 20](#).

For a more specific list of tasks involved in the runtime-based deployment process, see ["Getting Started with Runtime-Based Deployment" on page 26](#).

Repository-Based Deployment

In repository-based deployment, you build the assets from your development environment or version control system (VCS) and store them on a *repository*. The repository is a specific directory structure to which you map Deployer. Deployer deploys the assets contained in the repository to target servers, target groups, or both.

You can use repository-based deployment to deploy assets from the following kinds of webMethods runtimes to target servers:

- Application Platform
- BPM Process Development
- Broker
- Business Rules
- EDA
- Event Server

Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

- Integration Server
- My webMethods Server
- Optimize
- Trading Networks
- Universal Messaging

Composites

Composites are compressed files that contain the definitions of the assets and their dependencies you build in a development environment. Each composite defines assets from one webMethods runtime type. You build the assets into the composite file during the build process and store them in a repository for deployment.

Deployer can use the assets from several different composites (and webMethods runtime types) to construct a deployment set. The assets in each composite file retain the same relationships they share in the development environment.

The build process that creates the composites for your assets also creates an *Asset Composite Definition Language (ACDL) descriptor* (descriptor) for each composite. The descriptor is an XML schema that serves as a manifest for each composite and describes all of the assets included in the composite file. Deployer reads the descriptor to determine which assets are present in each composite.

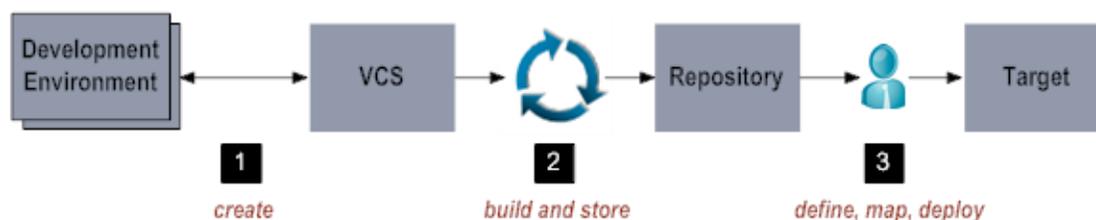
Build Script

In repository-based deployment, you use a *build script* (build.xml) to build composites and their associated descriptors from the development environment. The build properties file (build.properties) contains the settings the build script uses to build the assets.

For more information about setting build settings and running the build script, see ["Building Composites for Repository-Based Deployment" on page 29](#).

Overview of Repository-Based Deployment

The following graphic illustrates the asset deployment process in repository-based deployment:



The repository-based deployment process involves the following basic stages:

- Stage 1** Create assets on your development environment.
- Developers create assets in the development environment and save them on a server, or check them into a version control system (VCS).
- Stage 2** Build the composites and descriptors.
- You use the master build script to build the composite and descriptor files from the assets. You store these files in the repository.
- Stage 3** You use Deployer to do the following:
1. Define a *deployment project* and *deployment sets*. You select user-created assets from the ACDL composite stored on the repository to identify the assets to include in the project. For more information about deployment projects, see ["Creating Projects" on page 17](#). For more information about deployment sets, see ["Deployment Sets" on page 18](#).
 2. Map the contents of the deployment project to target servers. For more information, see ["Mapping Projects" on page 19](#).
 3. Deploy the assets in the project build to the target servers. For more information, see ["Deploying Projects" on page 20](#).

For a more specific list of tasks you perform in the repository-based deployment process, see ["Getting Started with Repository-Based Deployment" on page 27](#).

Creating Projects

A *deployment project* identifies the user-created assets on source servers (for runtime-based deployment) or a repository (for repository-based deployment) that you want to deploy to target servers. To create a project, you assign the project a name and set its properties, and then you authorize users to perform the project tasks.

When you create a project, Deployer automatically creates an HTML home page for the project. You can modify this page to contain instructions or notes about the project that you want users to view. For example, you might want to list the target servers for the users who will perform the mapping task, or you might want to provide instructions for users who will test the deployed solution.

Deployment Sets

You identify the assets to include in the project using *deployment sets*. Each deployment set identifies the user-created assets you want to deploy from one type of source server or repository to a target server.

A single project can include deployment sets with assets from different types of webMethods applications. For example, a single project can include Integration Server deployment sets and ProcessModel deployment sets.

Note: A project can contain deployment sets whose assets were selected from only one of the following:

- Source webMethods runtime types (for runtime-based deployment)
- A composite in a repository (for repository-based deployment)

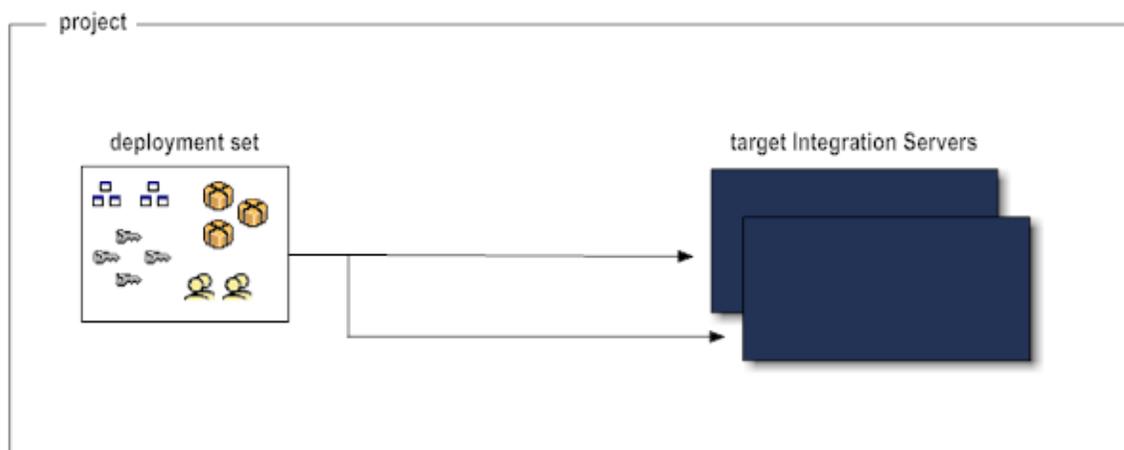
A project cannot contain both types of deployment sets.

Unresolved Dependencies

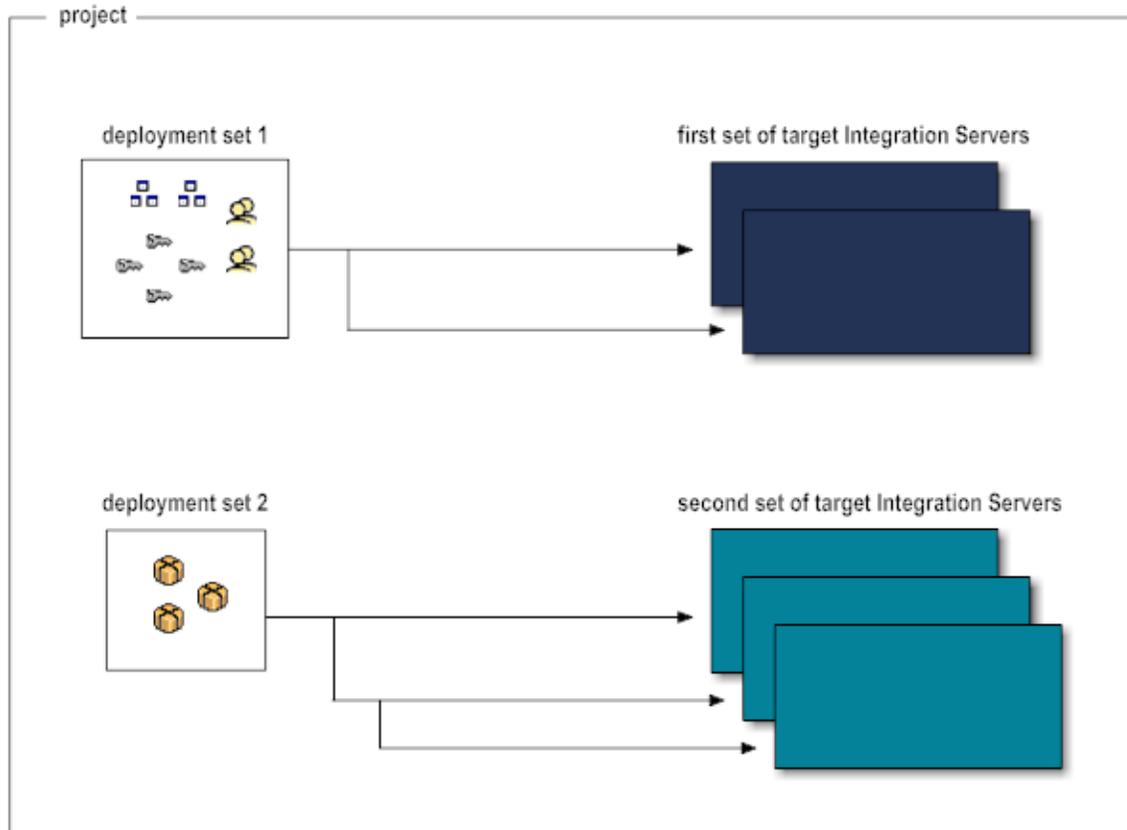
If assets in your deployment set depend on assets that are not part of the deployment set, Deployer identifies these missing assets as *unresolved dependencies*. For example, if you add a trigger to an Integration Server deployment set, but do not add the service that is invoked by the trigger, Deployer identifies the missing service as an unresolved dependency. Deployer enables you to resolve unresolved dependencies.

Target Servers

To control to which targets, the assets are deployed, you define target servers. You define target servers according to the kind of assets you want to deploy. For example, if you want to deploy Integration Server assets to one set of target Integration Servers, you can define a single deployment set that identifies those assets.



To deploy some Integration Server assets to one set of target Integration Servers and other Integration Server assets to a second set of target Integration Servers, you must define two different deployment sets.



Deletion Sets

Deployment projects that contain deployment sets whose assets were selected from source webMethods servers or repositories can also contain a *deletion set* for each target server. A deletion set lets you specify user-created assets to delete from the target server before you deploy the assets in the project's deployment sets. You can also export deletion set definitions from one project and import them into another.

Mapping Projects

In a *deployment map*, you identify target servers and target groups for each deployment set in a project. You can create multiple deployment maps for each project build (for example, if you are deploying to multiple environments).

Deploying Projects

To deploy a project, you first create a *deployment candidate*, which is a combination of a deployment set and a deployment map. For runtime-based deployment projects, the deployment candidate also includes the project build. Deployer does the following when you deploy the project:

- For projects that include deletion sets, Deployer deletes the identified assets from the target servers.
- Copies the contents of each deployment set in the deployment candidate's project to the target servers identified in the deployment candidate's deployment map.
- Creates a *deployment report* that lists all actions that occurred during deployment.
- For an IS & TN deployment set for which you specified substitute configuration values, Deployer substitutes those values during deployment.

Checkpoint and Roll Back

If a deployment to a target server fails and the target environment is in an inconsistent state, or a deployment is successful but the deployed assets are not working as expected, you can use Deployer's roll back feature to undo the deployment.

For deployment sets, you create a *checkpoint* to which you want to roll back the target server *before* you deploy. The checkpoint contains a copy of the assets on the target server that will be replaced by the assets in the deployment sets.

The roll back feature rolls back the target server to the checkpoint. If the deployment sets added assets to the target servers, the roll back removes them.

You can set Deployer to automatically create a checkpoint and roll back deployment sets when deployment fails. For deletion sets, Deployer automatically creates the checkpoint for the target server. If deployment fails, Deployer automatically rolls back the target server.

For more information about setting project checkpoints and roll back settings, see ["Setting General Deployment Defaults" on page 79](#).

Transactional Deployment

For repository-based deployment, you can deploy assets and composites using *transactional deployment*. When transactional deployment is enabled, Deployer automatically creates the checkpoint for the target server when you deploy the deployment set. If deployment fails, Deployer automatically rolls back all of the target servers to the state they were in when Deployer created the checkpoint. For more information about setting transactional deployment, see ["Creating a Project" on page 85](#).

Concurrent and Sequential Deployment

Deployer supports both sequential and concurrent asset deployment.

In a *concurrent* deployment, Deployer uses the host Integration Server thread pool to create a separate thread in order to deploy the assets to all target servers simultaneously. This is the default setting. You should configure the minimum and maximum threads in the pool to accommodate your system. For more information about setting the minimum and maximum thread pools, see *webMethods Integration Server Administrator's Guide*. Deployer writes records to the Audit Log to indicate that the main deployment thread has created new deployment threads for the specific target server. This allows you to track which thread belongs to each deployment request.

In a *sequential* deployment, Deployer deploys the assets to all target servers one by one in the order that the targets were added to the deployment map. If the project is not set to use concurrent deployment, Deployer deploys assets sequentially.

You can set concurrent deployment for all projects in the system when you set the default settings for Deployer as described in ["Setting General Deployment Defaults" on page 79](#). If the default setting is set to sequential deployment and you want to use concurrent deployment for a specific project (or vice versa), you can override the default setting for that project during creation. For more information about overriding the default settings for a project, see ["Creating a Project" on page 85](#).

Deployer Interfaces

Deployer offers a graphical user interface (GUI), a command line interface, and a Project Automator tool. Using the command line interface you can enter commands at a command prompt or you can create scripts that execute commands automatically. The Project Automator allows you to specify project information in an XML file for automated project creation.

The table below shows which tasks you can perform from each type of Deployer interface.

Task	GUI	Command Line	Project Automator
Add and view instructions or notes about projects.	X		
Authorize groups to work on projects.	X		
Configure communication between Deployer and the source servers or repositories and target servers.	X		X

Task	GUI	Command Line	Project Automator
Create project builds.	X	X	X
Create projects.	X		X
Define deployment and deletion sets.	X		X
Delete projects and set default properties for projects	X	X	
Export and import project properties and deletion set definitions.	X	X	
Generate a checkpoint or roll back target servers, list and display rollback reports.	X	X	
Generate progress reports.	X		
List, create, display, or delete deployment candidates; simulate deployments; deploy; list and display simulation and deployment reports.	X	X	
List, display, export, import, and delete deployment maps.	X	X	
List, export, and import builds; display build contents; list and display build reports.	X	X	
Map deployment and deletion sets to target servers.	X		X
Select assets from an ACDL descriptor file	X	X	X

Automating Project Creation

Deployer enables you to automate the following:

- Creation of projects.
- Definition of aliases for source servers and repositories and target servers.

- Creation of deployment and deletion sets.
- Creation of builds, maps, and deployment candidates.

You provide the necessary specifications in an XML file, then run Deployer's Project Automator tool. Optionally, you can export projects created in the GUI to an XML file. For more information about the Project Automator tool, see "[Automating Project Creation](#)" on page 187.

Logging

Deployer writes audit log entries to these logs:

- The Deployer GUI audit log. This log contains information about actions that users perform through the Deployer GUI, such as creating builds and deploying.
- The Deployer command line audit log. This log contains error messages written by Deployer commands executed by users.
- The Deployer Project Automator audit log. This log contains information and error messages written by the Project Automator during execution of an XML file.

Deployer writes journal entries to the Integration Server server log. The 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 Integration Server packages such as Deployer. For complete information about the Integration Server server log, see *webMethods Integration Server Administrator's Guide*.

2 Getting Started

- Getting Started with Runtime-Based Deployment 26
- Getting Started with Repository-Based Deployment 27

Getting Started with Runtime-Based Deployment

You will perform the following general tasks to deploy assets in a runtime-based deployment scenario:

Task	See...
Start Deployer.	"Starting Deployer" on page 54
Set up connections to source and target servers.	<ul style="list-style-type: none"> <li data-bbox="716 646 1351 720">■ "Connecting to Integration Servers and Trading Networks Servers" on page 56 <li data-bbox="716 730 1351 804">■ "Connecting to Optimize Servers" on page 57 <li data-bbox="716 814 1351 888">■ "Connecting to My webMethods Servers" on page 59 <li data-bbox="716 898 1351 951">■ "Connecting to Event Servers" on page 62 <li data-bbox="716 961 1351 1014">■ "Connect to Broker Servers" on page 63 <li data-bbox="716 1024 1351 1098">■ "Connect to BPM Process Model Servers" on page 65 <li data-bbox="716 1108 1351 1182">■ "Connecting to Business Rules Integration Servers" on page 70 <li data-bbox="716 1192 1351 1234">■ "Creating Target Groups" on page 73
Create a project.	"Creating and Managing Projects" on page 77
Define deployment set.	"Defining a Deployment Set" on page 95
Add assets to the deployment set.	"Adding Assets for Runtime-Based Deployment" on page 101
Define deletion set (optional).	"Defining a Deletion Set" on page 127
Build the project.	"Building a Runtime-Based Deployment Project" on page 137
Map assets to target servers and target groups.	"Mapping a Project" on page 141

Task	See...
Deploy the project.	"Deploying a Project" on page 151

Getting Started with Repository-Based Deployment

You will perform the following general tasks to deploy assets in a repository-based deployment scenario:

Task	See...
Build composites on your repository.	"Building Composites for Repository-Based Deployment" on page 29
Start Deployer.	"Starting Deployer" on page 54
Set up connections to the source repository and target servers.	<ul style="list-style-type: none"> ■ "Connecting to a Repository for Repository-Based Deployment" on page 72 ■ "Connecting to Integration Servers and Trading Networks Servers" on page 56 ■ "Connecting to Optimize Servers" on page 57 ■ "Connecting to Application Platform Servers" on page 58 ■ "Connecting to My webMethods Servers" on page 59 ■ "Connecting to Event Servers" on page 62 ■ "Connect to Broker Servers" on page 63 ■ "Connect to BPM Process Model Servers" on page 65 ■ "Connecting to Business Rules Integration Servers" on page 70
Create a project.	"Creating and Managing Projects" on page 77
Define a deployment and deletion sets.	<ul style="list-style-type: none"> ■ "Defining a Deployment Set" on page 95 ■ "Defining a Deletion Set" on page 127

Task	See...
Add assets to the deployment set.	"Adding Assets for Repository-Based Deployment" on page 117
Map assets to target servers or target groups.	"Mapping a Project" on page 141
Deploy the project.	"Deploying a Project" on page 151

3 Building Composites for Repository-Based Deployment

■ Overview	30
■ Before Building Composites	30
■ Setting the Properties for the Build	32
■ Running the Build Script and Rebuilding the Index	41
■ Preparing BPM Assets for Repository-Based Deployment	46
■ Next Steps	51

Overview

The build process for repository-based deployment involves the following basic stages:

Stage 1 Create the assets and export or copy them from the source development platform to the source directory.

Developers create the assets on the development environment and export or copy them to a file system or version control system (VCS) from which the build process can obtain them. This guide assumes that the assets have already been created. For more information about creating and exporting assets, see the documentation for the applicable webMethods runtime. For more information about adding assets to the source directory, see ["Adding Assets to the Source Directory" on page 31](#).

Stage 2 Install the webMethods Asset Build Environment on the build machine.

The Asset Build Environment supplies the scripts and libraries necessary to build assets into composites that Deployer then deploys to target servers and groups. For more information about the Asset Build Environment, see ["Installing the Asset Build Environment" on page 31](#). For information about installing the Asset Build Environment, see *Installing Software AG Products*.

Stage 3 Specify the parameters of the build.

Modify the build.properties file to define the parameters used by the master build script to build composites. For more information, see ["Setting the Properties for the Build" on page 32](#).

Stage 4 Run the master build script.

You run the master build script to package your assets into composites for deployment. For more information, see ["Running the Build Script and Rebuilding the Index" on page 41](#).

Note: If you are building a single BPM process project folder, follow the procedure described in ["Preparing the BPM Process Development Asset Deployment Environment" on page 48](#).

Before Building Composites

Building composites for repository-based deployment requires that you do the following:

- Create the assets you want to deploy and add them to a file system or VCS that is accessible to build scripts.
- Install the webMethods Asset Build Environment version 8.2 SP1 or greater.

Adding Assets to the Source Directory

In repository-based deployment scenarios, you create assets with a webMethods runtime and store them in a file system or version control system (VCS). The directory of the file system or VCS in which you store the assets is the *source directory*.

Most webMethods runtimes supply a means of exporting or saving the assets you create to the source directory. Other runtimes require you to manually copy the assets from one directory to another. For example, you can use My webMethods Server to export Broker assets to the source directory. However, to add *most* Integration Server administrative assets to the source directory, you must manually copy or check in the *Integration Server_directory/config* folder to the source directory. For a complete list of files and directories to copy or check in to the source directory for Integration Server administrative assets, see ["Adding Administrative Assets to the Source Directory" on page 258](#).

When specifying the properties for the build.properties file, you use the build.source.dir parameter to specify the full path of each source directory. Then, when you run the build script, it packages the contents of the source directories into the correct composite file format for deploying those assets to target servers. For more information about specifying parameters for the build.properties file, see ["Setting the Properties for the Build" on page 32](#). For more information about the composite files created the build script, see ["Running the Build Script and Rebuilding the Index" on page 41](#).

Keep the following points in mind while adding assets to the source directory:

- The source directory must be accessible to the build script.
- If the source directory is a file system on a VCS, you must supply the proper credentials for accessing and checking out the composite files from the source directory. For more information about supplying the VCS checkout properties, see ["Setting VCS Checkout Properties" on page 39](#).

Installing the Asset Build Environment

To build composites for deployment, you must first install the webMethods Asset Build Environment. The Asset Build Environment installs the build script and build properties file that you use to build deployment composites for webMethods runtimes.

For testing purposes, you can install the Asset Build Environment on the computer that you use to create assets in the development environment. For daily or continuous builds, you should install the Asset Build Environment on a separate computer, known as a *build machine*. The build machine is the computer on which you build the assets into components that Deployer can deploy to target servers.

For information about installing the Asset Build Environment, see *Installing Software AG Products*.

Setting the Properties for the Build

The Asset Build Environment installs the following files that you can customize to provide the build script the properties it needs to build composites for repository-based deployment:

- **build.properties.** Set the properties in this file before running the build script. For more information about setting the properties in `build.properties`, see ["Setting Build Properties" on page 32](#).
- **build-number.xml.** The build script uses this file to automatically version the assets included in the build incrementally. You can customize the file to generate the build number as needed (example, to match the VCS revision number).
- **build-source-checkout.xml.** If the build source directory is a version control system, you can use this file as a template to create an Ant task that checks out sources from a version control system (VCS). For information about setting the properties in `build-source-checkout.xml`, see ["Setting VCS Checkout Properties" on page 39](#).
- **build.bat or build.sh.** When building assets that are large in number or size, you should increase the Java memory available to the build process. For more information about adjusting Java memory settings, see ["Changing JVM Memory Settings" on page 40](#).

Note: When you *overinstall* later releases of the Asset Build Environment, your changes to these files are retained. You do not need to move the files unless you uninstall and reinstall the Asset Build Environment.

The Software AG Installer installs all three files in the following location as part of the Asset Build Environment:

`Software AG_directory\common\AssetBuildEnvironment\master_build`

Setting Build Properties

The master properties file, `build.properties`, controls the build settings for the repository-based build process. It contains a set of switches that enable and disable the build tasks for the corresponding runtime types in the build.

To set the build properties

1. Open the following file in a text editor:

```
Software AG_directory\common\AssetBuildEnvironment\master_build
\build.properties
```

2. Set the following properties:

<u>Property</u>	<u>Definition</u>																						
sag.install.dir	Path of the directory where Software AG products are installed.																						
build.output.dir	<p>Root directory where the build script will place the output (composites and descriptors) of the build. Use a forward slash "/" as path separator.</p> <p>The build script creates a subdirectory with the current build number in this directory, as well as a subdirectory for the composites and descriptors for each runtime type included in the build as follows:</p> <table border="1"> <thead> <tr> <th><u>If your build contains assets from...</u></th> <th><u>The build script creates the following subdirectory...</u></th> </tr> </thead> <tbody> <tr> <td>Application Platform</td> <td>ApplicationPlatform</td> </tr> <tr> <td>BPM Process Development</td> <td>BPM</td> </tr> <tr> <td>Broker</td> <td>Broker</td> </tr> <tr> <td>Business Rules</td> <td>Rules</td> </tr> <tr> <td>EDA</td> <td>EDA</td> </tr> <tr> <td>Integration Server</td> <td>IS</td> </tr> <tr> <td>My webMethods Server</td> <td>MWS</td> </tr> <tr> <td>Optimize</td> <td>Optimize</td> </tr> <tr> <td>Trading Networks</td> <td>TN</td> </tr> <tr> <td>Universal Messaging</td> <td>UniversalMessaging</td> </tr> </tbody> </table>	<u>If your build contains assets from...</u>	<u>The build script creates the following subdirectory...</u>	Application Platform	ApplicationPlatform	BPM Process Development	BPM	Broker	Broker	Business Rules	Rules	EDA	EDA	Integration Server	IS	My webMethods Server	MWS	Optimize	Optimize	Trading Networks	TN	Universal Messaging	UniversalMessaging
<u>If your build contains assets from...</u>	<u>The build script creates the following subdirectory...</u>																						
Application Platform	ApplicationPlatform																						
BPM Process Development	BPM																						
Broker	Broker																						
Business Rules	Rules																						
EDA	EDA																						
Integration Server	IS																						
My webMethods Server	MWS																						
Optimize	Optimize																						
Trading Networks	TN																						
Universal Messaging	UniversalMessaging																						
build.source.dir	Separated list of the full paths of the source directories that contain assets to build. All directories in the list must contain project directories (IS packages, CAF projects, TN exports, and so on) as direct children. Use																						

<u>Property</u>	<u>Definition</u>
	<p>“;” as the delimiter for more than one folder. Use a forward slash “/” as path separator.</p> <p>For example, if you want to include Integration Server and Trading Networks assets, you would enter the following:</p> <pre><i>/root_path /project_name /IS;/root_path /project_name /TN</i></pre> <p>Where <i>root_path</i> is the root directory of the source, <i>project_name /IS</i> is the directory holding Integration Server assets, and <i>project_name /TN</i> is the directory holding Trading Networks assets.</p>
<u>If your source directory contains assets from...</u>	<u>Input source directory naming convention or any restrictions...</u>
Application Platform	No naming conventions or restrictions.
BPM Process Development	No naming conventions or restrictions.
Broker	<i>/root_path/ project_name/Broker</i>
Business Rules	No naming conventions or restrictions.
EDA	No naming conventions or restrictions.
Integration Server	Refer " Adding Administrative Assets to the Source Directory " on page 258 and " Adding Package Assets to the Source Directory " on page 297.
My webMethods Server	No naming conventions or restrictions.

Property	Definition
	<p>Optimize No naming conventions or restrictions.</p> <p>Trading Networks No naming conventions or restrictions.</p> <p>Universal Messaging <code>/root_path/ project_name/UniversalMessaging</code></p>
<code>build.source.project.dir</code>	<p>Separated list of project folders containing the assets that the build script will build. All folders in the list must contain project folders only. This property is different from the <code>build.source.dir</code> as it defines individual project directories</p> <p>Use ";" as delimiter for more than one folder. Use a forward slash "/" as path separator.</p> <p>Note: If you are building an Integration Server project and specify a value for this property, you must also specify a value for <code>is.acdl.config.dir</code>.</p>
<code>enable.archive</code>	<p>Specifies whether the build script archives the output directory. If you set this field to <code>true</code>, you must provide a value for <code>build.archive.dir</code>.</p>
<code>build.archive.dir</code>	<p>Required if <code>enable.archive</code> is set to <code>true</code>. Location of the archive directory for the output. The build script creates a new subdirectory with the name of the build number. It then moves the contents of the output directory (set in <code>build.output.dir</code>) to this newly created directory.</p>
<code>build.version</code>	<p>Version number of the current build. The build script appends an automatically-generated incremental build number to this property to get the final build number.</p> <p>For example, if you set this property to <code>8.2</code>, the generated build number for the first build is <code>8.2.1</code> and the build number for the next build is <code>8.2.2</code>.</p>
<code>enable.checkout</code>	<p>Specifies whether the build script should check out the source files from a VCS.</p>

Property	Definition
	<ul style="list-style-type: none"> ■ To enable the check out task, set to <code>true</code>. The build script invokes the default target you set in <code>build-source-checkout.xml</code>. ■ To disable check out task, set to <code>false</code>.
	<p>Note: If this property is set to <code>true</code>, you must configure the properties in <code>build-source-checkout.xml</code> to synchronize with your VCS system. For more information, see "Setting VCS Checkout Properties" on page 39.</p>
enable.build	<p>Specifies the runtime types for which the build script should build composites. Each runtime type has a separate property. Set to:</p> <ul style="list-style-type: none"> ■ <code>true</code> to build composites for the specified runtime type. ■ <code>false</code> to exclude the specified runtime type.
Set this property...	To build composites for assets of this type...
enable.build. ApplicationPlatform	Application Platform
enable.build.IS	Integration Server
enable.build.MWS	My webMethods Server
enable.build.BPM	BPM Process Development
enable.build.TN	Trading Networks
enable.build.Optimize	Optimize
enable.build.Broker	Broker
enable.build.EDA	EDA
enable.build.RULES	Business rules
enable.build.	Universal Messaging

Property	Definition												
	UniversalMessaging												
build.log.enable	Specifies whether to enable logging for the build. Set to: <ul style="list-style-type: none"> ■ true to enable logging. ■ false to disable logging. 												
build.log.fileName	Name of the log file. The default file name is build.log. The build script creates the log file in the following directory: <p><i>Software AG_directory</i>\common \AssetBuildEnvironment\master_build</p>												
build.logLevel	Logging level to include in the log. Possible values are: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Specify...</th> <th>To record...</th> </tr> </thead> <tbody> <tr> <td>debug</td> <td>Debug, informational, warning, error, and fatal messages.</td> </tr> <tr> <td>error</td> <td>Error and fatal messages.</td> </tr> <tr> <td>info (default)</td> <td>Informational, warning, error, and fatal messages.</td> </tr> <tr> <td>verbose</td> <td>No possible values defined for log level.</td> </tr> <tr> <td>warn</td> <td>Warning, error, and fatal messages.</td> </tr> </tbody> </table>	Specify...	To record...	debug	Debug, informational, warning, error, and fatal messages.	error	Error and fatal messages.	info (default)	Informational, warning, error, and fatal messages.	verbose	No possible values defined for log level.	warn	Warning, error, and fatal messages.
Specify...	To record...												
debug	Debug, informational, warning, error, and fatal messages.												
error	Error and fatal messages.												
info (default)	Informational, warning, error, and fatal messages.												
verbose	No possible values defined for log level.												
warn	Warning, error, and fatal messages.												

3. If you are building BPM Process Development assets, set the following BPM-specific properties:

Note: See "[Preparing BPM Assets for Repository-Based Deployment](#)" on page 46 for BPM project-level behaviors and instructions.

Property	Definition
<code>bpm.acdl.model.ids</code>	<p>Optional. Semicolon-separated list of process IDs (that is, a concatenation of process project name and process model file name).</p> <p>If the process does not exist in the source directory or one of its subdirectories, the value is ignored. If left empty, all process model IDs are included. This property can be used with or without the <code>bpm.acdl.model.version</code> property.</p>
<code>bpm.acdl.bam.model.ids</code>	<p>Optional. Semicolon-separated list of BAM process model IDs to add to the composite. BAM process models are tracking only and do not declare any dependencies. For example: <code>testProcessProject/testProcessModel;testProcessProject2/testProcessModel2</code>. If the process does not exist in the source directory or a child directory of it, the value is ignored. If left empty, no process model IDs are included. All process models are assumed to be BPM processes and will generate the standard process dependencies. This property can be used with or without the <code>bpm.acdl.model.ids</code> or <code>bpm.acdl.model.version</code> properties.</p>
<code>bpm.acdl.model.version</code>	<p>Optional. Version number of the process models to include in the build. Enter a single integer value (example, <code>1</code>).</p> <p>If this value is set, the build script includes only those process models in the source directory or one of its subdirectories that match the version number. If left empty, the build script includes process models of all versions. You can use this property with or without the <code>bpm.acdl.model.ids</code> property.</p>

4. If you are building Optimize assets, set the following Optimize-specific property:

Property	Definition
<code>optimize.acdl.validation</code>	<p>Specifies whether to switch the XSD validation of the generated composite and descriptor files. Set to:</p>

Property	Definition
	<ul style="list-style-type: none"> ■ <code>On</code> to use XSD validation. ■ <code>Off</code> to turn off XSD validation. This is the default.

5. If you are building business rules assets, set the following business rules-specific property:

Property	Definition
<code>RULES.skip.on.validation.warning</code>	<p>Specifies whether the build script should create a rule project archive and composite when the rule project issues validation warnings. Set to:</p> <ul style="list-style-type: none"> ■ <code>true</code> to create a project archive and composite. ■ <code>false</code> to turn off project archive and composite creation. This is the default.

6. If you are building Integration Server assets, set the following Integration Server-specific property:

Property	Definition
<code>is.acdl.config.dir</code>	<p>Specifies the full path of the Integration Server config directory. For example:</p> <p><i>Software AG_directory</i>\IntegrationServer\config</p> <p>Note: If you set this property, you must also specify a value for <code>build.source.project.dir</code>. If you do not specify a value for <code>build.source.project.dir</code>, the build script ignores this value.</p>

7. Save and close the file.

Setting VCS Checkout Properties

If you set `enable.checkout` in `build.properties` to `true`, configure the properties in `build-source-checkout.xml` to check out the assets from a VCS.

To set VCS checkout properties

1. Open the following file in a text editor:

Software AG_directory\common\AssetBuildEnvironment\master_build\build-source-checkout.xml

2. Set the following properties to correspond to your VCS as follows:

Note: The build-source-checkout.xml template is specific to an SVN version control system. You might have to add more properties to make the file work with other VCS types.

Property	Definition
svn.jars.dir	Location of the open source SvnAnt libraries (http://subclipse.tigris.org/svnant.html). These libraries are required by the checkout Ant task. To use a VCS other than SVN, modify this parameter accordingly.
svn.user	User name of the SVN.
svn.password	Password of the SVN.
svn.url	URL of the SVN from where the build script checks out the assets.
build.checkout.dir	The root directory from which the build script will check out the asset sources. This property can point to only one directory; therefore, it should be the root directory containing <i>all</i> projects that will be built. For example, if your solution contains an Integration Server project, your full project directory might be /root/project/builds/webM/IS. In this case, you would not include /IS in the path. Instead, you would enter /root/project/src/webM/.

3. Save and close the file.

Note: Do not rename or remove the file from the master_build directory.

Changing JVM Memory Settings

You can adjust the JVM memory settings available to the build. This is helpful when building a large number of assets or assets that contain a large amount of data.

To change the JVM memory settings

- Using a text editor, open the following file:

<u>For this platform...</u>	<u>Open the following file...</u>
Windows	<i>Software AG_directory</i> \common\AssetBuildEnvironment\ bin\build.bat
UNIX	<i>Software AG_directory</i> /common/AssetBuildEnvironment/ bin/build.sh

- Change the values of the following parameters as necessary:
 - `JAVA_MAX_MEM`
 - `JAVA_MAX_PERM_SIZE`
- Save and close the file.

Running the Build Script and Rebuilding the Index

The build script uses the properties you specified for the `build.properties` file in "[Setting the Properties for the Build](#)" on page 32 to build the assets created in the development environment into composites. Specifically, the build script:

- **Checks out the asset sources.** The build script can use a VCS to obtain the asset sources to build into composites. If you use a VCS to store your assets, you must set the `enable.checkout` property in `build.properties` to `true` and set the properties in `build-source-checkout.xml` to access the VCS. For more information, see "[Setting Build Properties](#)" on page 32 and "[Setting VCS Checkout Properties](#)" on page 39.
- **Creates and indexes the repository.** The build script creates the repository in the specified structure. This is the source from which Deployer selects the assets for deployment. For more information about defining a repository in Deployer, see "[Connecting to a Repository for Repository-Based Deployment](#)" on page 72.
- **Versions the assets.** The build script uses the `build-number.xml` file to automatically version the assets included in the build in the format `version_number.build_number`.
- **Builds the composites and descriptors in the repository.** The build script generates the following files for each runtime type in the build:
 - ACDL descriptor file in the format `project_name.acdl`
This is the descriptor file that contains the metadata that lists the assets and dependencies contained in the composite.

- A compressed composite file that contains the definitions of the assets and their dependencies for deployment. The composite file is in the format *project_name.composite_file_type* where:

For this runtime type...	<i>project_name</i> is the...	<i>composite_file_type</i> is...
Application Platform	When the project is a standard web application, its name is used as <i>project_name.war</i> .	.war
	When the project is an OSGi bundle, its name is used as <i>project_name.jar</i> .	.jar
	When the project is an OSGi web bundle, its name is set to <i>project_name.jar</i> .	.jar

Note: The difference between bundle and Web bundle is in the jar content. Web bundles are valid OSGi bundles and are structured internally as .war files, that is, they are a fusion of the first two project types. The difference is also reflected in the "acdl" file of the web bundles. They contain both an asset of type "Bundle" and an asset of type "WebApp".

The fourth type of composite is configuration files. They have the file extension as ".properties". They do not have their own projects. Instead they are co-hosted in the */src/main/config/* directory of either a war or a bundle or a web bundle project. The name of the host project is not used to build the name of the configuration composite. These files are "built" simply by copying them to the asset repository. Therefore they are named "property_file_name.properties". An additional file "property_file_name.acdl" is created to make them into individually deployable assets. It is possible to create a project that constitutes only a collection of configurations in */src/main/config*. These configurations can customize either other composites or the OSGi Common Platform itself.

For this runtime type...	<i>project_name</i> is the...	<i>composite_file_type</i> is...
BPM process model	Process ID with “/” replaced by “_”. For more information about process models, see <i>webMethods BPM Process Development Help</i> .	.process
Broker	<p>Name of the file you exported using My webMethods Server prefixed with “Broker_” or “Provider_” depending on whether you exported Broker assets or JNDI assets, respectively.</p> <p>When you export Broker assets (document, client, or client group), Broker creates an ADL file for the Broker assets.</p> <p>To export assets from webMethods Broker 8.2 SP2 or earlier, you must migrate the assets from XML format to ADL format for use in Deployer. For more information about migrating Broker assets from XML to ADL format, see <i>Upgrading Software AG Products</i>.</p> <p>When you export JMS destinations (JMS queues and JMS topics) created by a webMethods JNDI provider, Broker creates a separate XML file for a JNDI context.</p> <p>For more information about exporting Broker assets, see <i>Administering webMethods Broker</i>.</p>	.adl (Broker assets) .xml (JNDI assets)
Business Rules	Name of the rules project.	.jar

For this runtime type...	<i>project_name</i> is the...	<i>composite_file_type</i> is...
EDA	Name of the Event Types project.	.zip (Event Type assets)
Integration Server	<ul style="list-style-type: none"> ■ For package, adapter runtime, and .Net assets, this is the name of the Integration Server package. For more information about adding package assets to the source directory for inclusion in the build, see "Adding Package Assets to the Source Directory" on page 297. For more information about adding adapter runtime or .Net assets to the source directory for inclusion in the build, see "Adding Adapter Runtime and .NET Service Assets to the Source Directory" on page 304. ■ For administrative assets, this is isconfiguration. For more information about adding administrative assets to the source directory for inclusion in the build, see "Adding Administrative Assets to the Source Directory" on page 258. 	.zip
My webMethods Server	Name of the project you exported. For more information about exporting My webMethods Server assets, see <i>Administering My webMethods Server</i> .	.skin, .war, .cdp
Optimize	Name of the file you exported using My webMethods Server. For more information about	.zip

For this runtime type...	<i>project_name</i> is the...	<i>composite_file_type</i> is...
	exporting Optimize assets, see <i>Administering webMethods Optimize</i> .	
Trading Networks	Name of the file you exported using My webMethods Server. For more information about exporting Trading Networks assets, see <i>webMethods Trading Networks Administrator's Guide</i> .	.bin
Universal Messaging	Name of the file you exported using Universal Messaging Enterprise Manager.	.xml

For example:

- The composite for a Broker project containing Broker assets would be named `Broker_file_name.adl`, where *file_name* is the name of the file you exported from Broker using My webMethods Server.
- The composite for a Broker project containing JNDI assets would be named `Provider_file_name.xml`, where *file_name* is the name of the file you exported from Broker using My webMethods Server.
- The composite for a project containing Integration Server administrative assets would be named `isconfiguration.zip`.

Deployer uses the composite and descriptor files to deploy the assets to a target server. For more information about deploying assets, see ["Adding Assets for Repository-Based Deployment" on page 117](#).

Note: Do not customize this file. When you overinstall later releases of the Asset Build Environment, your changes will be lost.

Running the Build Script

Before you run the build script, make sure you have set the properties in the build properties file. For more information, see ["Setting the Properties for the Build" on page 32](#).

To run the build script

1. Run one of the following commands from the *Software AG_directory* \common \AssetBuildEnvironment\bin directory:

<u>For this platform...</u>	<u>Run the following command...</u>
Windows	build.bat
UNIX	build.sh

The build script builds the composite files and places them in the location specified by the build.output.dir property you specified in ["Setting the Properties for the Build" on page 32](#).

Rebuilding the Index

If the index you created becomes corrupted or the repository index is accidentally deleted from the repository, you can recreate the index without rebuilding the entire repository. By default, when you use this procedure, Deployer rebuilds the index specified in the **File Directory** box. For more information about the **File Directory** box, see ["Connecting to a Repository for Repository-Based Deployment" on page 72](#).

When you follow this procedure to rebuild the index, Deployer rebuilds *only* the index. To build the index, check out the asset sources, version the assets, and build the composites and descriptors in the repository, you must run the build script as described in ["Running the Build Script" on page 45](#).

To rebuild the index

1. In Deployer, go to the **Deployer > Repository** page.
2. If you want to change the repository directory, perform the following:
 - a. In the **Name** column, click the name of the repository to edit.
Deployer opens the repository properties in the right-hand pane.
 - b. In the **File Directory** box, type the full path of the repository for which to rebuild the index.
 - c. Click **Save Changes**.
3. In the **Create Index** column for the repository, click .

Preparing BPM Assets for Repository-Based Deployment

This section describes how to configure and run a build script that creates a composite and ACDL descriptor for specified process models. Deployer uses the resulting

composite and descriptor files to deploy the process development assets as part of a deployment project.

Differences Between Manual Process Generation and Deployed Processes

There are distinct differences in the results when you generate a process model manually in Software AG Designer and when you deploy a process model with Deployer.

About the Deployment of Generation Receipts

Deployer does not deploy generation receipts when deploying Designer process models. The primary reason for this is that the generation receipt stores a logical-to-physical server mapping and the only information available to Deployer is in the composite file, which does not contain any mapping information.

When you regenerate a process, the regeneration process uses the mapping information to determine whether a logical server mapping has changed from the previous generation. If the mapping has changed, the regeneration process connects to the previous logical server and cleans up the process-related assets there (for example, deleting associated triggers and services), in addition to creating the new assets on the new logical server. With no generation receipt, this mapping information is not available and the clean-up procedure cannot occur.

Note: If a user connects Designer to a target environment and regenerates a deployed process to servers other than those deployed to the original process, the wrapper services and triggers on the original servers are not deleted. Wrapper services and triggers are still generated on the new logical servers, and existing services and triggers on unchanged logical servers remain unchanged. You must perform a manual cleanup on old logical servers. If you do not, multiple triggers and services could run in the process.

If you regenerate a previously deployed process to the same logical servers, Designer creates a new generation receipt so that future regeneration processes are handled correctly.

About the Redeployment of Processes

Because repository-based deployment does not generate a generation receipt, when you deploy a process to a target environment using repository-based deployment, you cannot then redeploy that process using that target environment as a source for a runtime-based deployment project. In runtime-based deployment Deployer requires the generation receipt to determine dependencies. If you want to redeploy the process, you must first rebuild the process in the target environment.

About the Deployment of Process Images

When you deploy a process model, the associated process model image and icon images are not deployed with the model. As a result, any webMethods Monitor APIs that use the process or icon images will not perform as expected with the deployed process.

Preparing the BPM Process Development Asset Deployment Environment

You prepare process development assets for deployment by creating a composite and descriptor file for each .process file to be deployed. You create these files by configuring and running the one of two build scripts as follows:

- The master-level build script you run in the \bin directory of your installed Asset Build Environment environment as described in ["Running the Build Script and Rebuilding the Index" on page 41](#). Use this build script for standard use in a defined deployment environment.

The build.properties file must be configured prior to running the build script. For information about configuring the build.properties file, see ["Setting the Properties for the Build" on page 32](#).

- The project-level build script, in which the build script is run in a process project directory. Running this build script is useful because process project directories can be located anywhere in a user's file system (that is, outside the deployment environment). This approach enables you to create the composite and descriptor files in any process project directory, regardless of location. After the composites and descriptors are built, they can be moved or copied into a deployment repository and indexed by the build script in order to make them available to Deployer. Project-level build script behavior is governed by the build.xml file in the process project directory. For information about configuring the build.xml file, see ["Configuring the Process Project build.xml File" on page 49](#).

About build.xml Files

Be aware that in either mode, the build script processes *only* those process projects that contain a build.xml file. Process projects without a build.xml file are ignored, even if a process model is explicitly named.

Software AG Designer adds a build.xml file to a process project automatically for projects created with Process Development version 8.2 SP1 and later. If a process project has no build.xml file, you can copy the build.xml file from any other project. If the build.xml was not modified in the originating project, you do not have to modify the build.xml file to copy it from one project to another.

Configuring the Process Project build.xml File

Note: There is no need to configure the build.xml file if the build script is being run in the master_build directory. Configuration of the build.xml file is an option only when you want to run the build script in a process project directory. For information about running the build script in the master_build directory, see ["Running the Build Script and Rebuilding the Index" on page 41](#).

A BPM build.xml file *must* be included in each process project (see ["About build.xml Files" on page 48](#)). The file contains the default properties that are used when the build script is run in the process project directory (that is, no properties are being passed into the project build from a script running in the master_build directory).

Note: A BPM build.xml file is included by default if the Designer process project is created in Designer 8.2 or later. If the process project is imported from previous versions of Designer, you must add it manually to the process project folder.

When the script is run in the process project directory, the build script assumes that the source directory and the output directory are the same as the current directory.

Note: The property settings described below are used *only* when the build script is running in the process project directory. If the build script is running in the master_build directory, the build.xml properties described below are overridden by the properties in the build.properties file.

To configure the BPM build.xml file

1. In the process project you want to work with, open the build.xml file in a text editor.
2. Define the following properties as needed:

Property	Definition
<code>default.sag.install.dir=path/to/directory</code>	Required. The location of the Software AG suite installation directory. The specified directory must contain the <code>common/lib</code> directory.
<code>sag.master.build.dir=path/to/directory</code>	Required. The location of the Deployer environment you installed with Software AG Installer. The specified directory must contain the <code>master_build</code> directory.
<code>default.bpm.acdl.model.ids=processID1;processID2</code>	Optional. A semicolon-separated list of process IDs (that is, a concatenation of

Property**Definition**

<code>default.bpm.acdl.model.version=n</code>	<p>Optional. A single integer value that is matched to the version number of process models in the source directory or a child directory of it. The build script includes only models with a version number equal to this value in the build. If left empty, the build script includes all process model versions. You can use this property with or without the <code>model.ids</code> property.</p>
<code>default.bpm.acdl.bam.model.ids</code>	<p>Optional. A semicolon-separated list of BAM process model IDs. For example: <code>testProcessProject/testProcessModel;testProcessProject2/testProcessModel2</code>.</p> <p>You must specify all BAM process models because they are for tracking purposes only and do not need to declare any dependencies. If the process does not exist in the source directory or a child directory of it, the build script ignores the property.</p> <p>If left empty, all process models are assumed to be BPM processes. The build script treats any process models it builds but that are not included in this list as BPM processes and generates the standard process dependencies for them. You can use this property with or without the <code>bpm.acdl.model.ids</code> or <code>bpm.acdl.model.version</code> properties.</p>

3. Save the file.

Running the Build Script in a Process Project Directory

Use this procedure if you want to run the build script on a single process project. In this case, the build script creates composite and descriptor files within the process project directory. The build script considers all process files in the process project directory and its sub-directories, as specified by the properties in build.xml.

To run the build script in a process project directory

1. Define any of the available build.xml properties as described in "[Configuring the Process Project build.xml File](#)" on page 49.
2. Open a command prompt window.
3. Change to the process project directory where you want to run the build script.
4. Run this command: `ant`

Note: Ensure that you have modified your system variables to contain the path name of the build script. Otherwise, you must type the full path name.

The build script processes the specified process model files and creates corresponding composite and descriptor files in the current directory. After the composite and descriptor files are built, you can move or copy the files to a deployment repository and index them for use with Deployer.

5. If you want to use the composite and descriptor files for use with Deployer, perform the following additional steps:
 - a. Move or copy the files to the BPM subdirectory of the repository.
 - b. Run the following command from the *Software AG_directory* \common \AssetBuildEnvironment\bin directory:

For this platform...

Run the following command...

Windows

`build.bat createIndex`

UNIX

`build.sh createIndex`

The `createIndex` script indexes the files you moved or copied to the repository.

Next Steps

Once you have built your composites, you can connect to the repository and target servers, create a project, and create a deployment set that includes the composites for deployment.

To...	See...
Connect to the repository and target servers	"Starting Deployer and Connecting to Servers" on page 53
Create a project	"Creating and Managing Projects" on page 77
To create a deployment set	"Defining a Deployment Set" on page 95.

4 Starting Deployer and Connecting to Servers

■ Starting Deployer	54
■ Connecting to webMethods Servers	54
■ Connecting to a Repository for Repository-Based Deployment	72
■ Editing Properties for Source and Target Servers	72
■ Creating Target Groups	73
■ Next Steps	76

Starting Deployer

Deployer starts automatically when you start its host Integration Server.

Open the Deployer interface by entering this URL in an Internet browser, where *Integration Server_port* is the host name and port of the Integration Server that hosts Deployer:

`http://Integration_Server_host:Integration_Server_port/WmDeployer`

Deployer and Integration Server use the same log on user name and password. If you just installed Deployer with a new Integration Server, the defaults are user name Administrator and password manage.

Connecting to webMethods Servers

You can perform the tasks in this section to connect to the source and target webMethods servers.

For runtime-based deployment projects, the *source* servers are those servers from which you select assets for deployment. For repository-based deployment you do not connect to source servers. The source for repository-based deployment is *always* the repository on which the composites are built. For more information about adding a source repository, see "[Connecting to a Repository for Repository-Based Deployment](#)" on page 72.

For both runtime- and repository-based deployment, you can perform the tasks in this section to map the assets from the source servers or repository to target servers. You define *target* servers to define the location to which Deployer deploys assets from the source server or repository.

For every source and target server you add, you select the version of the webMethods runtime for that server. For example, if you are adding a target Integration Server that is running version 8.2 SP1, you would select 8.2 for the version. The version you select affects the source and target servers you can include in the project as follows:

- For runtime-based deployment projects, after you select the source servers, Deployer displays only servers running compatible versions for selection as target servers. For example, if you select source servers of version 8.0, Deployer displays only target servers and target groups with a version of 8.0 or 8.2 for mapping.

The following table lists all version compatibilities:

<u>Source Version</u>	<u>Compatible Target Versions</u>
7.1	7.1

Source Version	Compatible Target Versions
8.0	8.0, 8.2 8.2 targets support only the following runtimes from 8.0 sources: <ul style="list-style-type: none"> ■ Integration Server ■ Trading Networks ■ BPM process models
8.2	8.2
9.0	9.0
9.5	9.5
9.6	9.6
9.7	9.7
9.8	9.8

Additionally, you can import builds from projects created with Deployer version 8.2 or earlier to Deployer 9.0 or later for deployment. For example, you could import a build from a version 8.0 project into Deployer 9.0.

- For repository-based deployment projects, each project includes sources and targets of only one version. You cannot include source repositories or target servers of different versions.
- For repository-based deployment projects, Deployer deploys only target servers and target groups of version 8.2 SP1 and later. Because repository-based deployment does not use source servers, Deployer limits the version of the targets in your project according to the first target server or target group you select for mapping. For example, if you select a target server of version 8.2, Deployer displays only target servers and target groups of version 8.2 for mapping.

You can connect to the following source and target webMethods servers:

To connect to...	See...
Integration Servers or Trading Networks servers	"Connecting to Integration Servers and Trading Networks Servers" on page 56

To connect to...	See...
Optimize servers	"Connecting to Optimize Servers" on page 57
Application Platform servers	"Connecting to Application Platform Servers" on page 58
My webMethods Servers	"Connecting to My webMethods Servers" on page 59
Event Servers	"Connecting to Event Servers" on page 62
Broker servers	"Connect to Broker Servers" on page 63
BPM process models	"Connect to BPM Process Model Servers" on page 65
Universal Messaging servers	"Connect to Universal Messaging Servers" on page 67
EDA deployment endpoints	"Connecting to EDA Deployment Endpoints" on page 69
Business Rules Integration Servers	"Connecting to Business Rules Integration Servers" on page 70
Business Rules My webMethods Servers	"Connecting to Business Rules My webMethods Servers" on page 71

Connecting to Integration Servers and Trading Networks Servers

All connections you create for Integration Servers and Trading Networks servers are *remote server aliases*. A remote server alias contains the connection information required to connect to a remote Integration Server or Trading Networks server. For more information about remote servers, and instructions on defining them, see *webMethods Integration Server Administrator's Guide*.

To connect to source and target Integration Servers and Trading Networks servers

1. In Deployer, go to the **Servers > IS & TN** page.
2. Click **Add Remote Server Alias**.

Deployer opens the Integration Server Administrator to the **Settings > Remote Servers > Create Alias** page of the Integration Server that hosts Deployer.

3. In Integration Server Administrator, define the remote servers by completing the fields in the **Remote Server Alias Properties** area as described in *webMethods Integration Server Administrator's Guide*. You should define the following as remote servers:
 - All source Integration Servers and Trading Networks servers.
 - All target Integration Servers and Trading Networks servers.
 - The Integration Server that hosts Deployer, if you will be using it as a source or target server (that is, define the Integration Server as a remote server to itself).
4. In Deployer, go to the **Servers > IS & TN** and click **Refresh this Page**.

Deployer displays the new server alias to the **Remote Servers List** area of the **Servers > IS & TN** page.

5. Select the version of the Integration Server hosting the source or target from the **Version** box.

For example, if the host Integration Server is running version 8.2 SP1, you would select **8.2**. For information about selecting the version, see "[Connecting to webMethods Servers](#)" on page 54.

6. Install the WmDeployerResource package on each Integration Server.

The WmDeployerResource package is the implementation of the operation endpoints for Integration Server and Trading Networks. Install the package as follows:

- a. In Deployer, go to the **Servers > IS & TN** page.

The page lists all Integration Servers you defined as remote servers.
 - b. In the **Install** column, select the check box next to each Integration Server on which you want to install the WmDeployerResource package.
 - c. Click **Install**.
7. Click **Save**.

Connecting to Optimize Servers

Use the following procedure to set up connections to Optimize servers.

To connect to source and target Optimize servers

1. In Deployer, go to the **Servers > Optimize** page.
2. For every source and target Optimize server, click **Configure Optimize Server**. In the **Configure Optimize Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server. Note: You cannot use "localhost" for Optimize servers.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Optimize server. For information about selecting the version, see "Connecting to webMethods Servers" on page 54.
Use SSL	Whether Deployer should use SSL to connect to the server.

- Click **Configure**. To test the connection, click .

Connecting to Application Platform Servers

The Application Platform is an add-on for existing products. You can add Application Platform only on top of Integration Server. Application Platform build needs a local installation of Integration Server > Application Platform Support to build the user projects.

Note: Application Platform supports only repository-based deployment, that is, it cannot serve as a source server. Currently, the Application PlatformDeployer Endpoint is added only to the Integration Server when Integration Server > Application Platform Support is installed. In future releases, other products will also acquire their specific Application Platform add-ons and the endpoint will be installed in their runtimes as well.

Use the following procedure to set up connections to the Application Platform servers.

To connect to source and target Application Platform servers

1. In Deployer, go to the **Servers > Application Platform** page.
2. For every source and target Application Platform server, click **Configure Application Platform Server**. In the **Configure Application Platform Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: <code>\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "</code>
Host	Host name or IP address of the Application Platform server.
Port	Port for the server. Note: For Integration Server, the port is that of the Integration Server Shared Platform Tomcat instance, rather than the default Integration Server HTTP port. The Shared Platform Tomcat Instance is distinct from the wmTomcat Integration Server Package. It is a separate Tomcat runtime used specifically by the Application Platform add-on.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Application Platform server. For information about selecting the version, see "Connecting to webMethods Servers" on page 54 .
Use SSL	Whether Deployer should use SSL to connect to the Application Platform server.

3. Click **Configure**. To test the connection, click .

Connecting to My webMethods Servers

Use the following procedure to set up connections to My webMethods Servers.

To connect to source and target My webMethods Servers

1. In Deployer, go to the **Servers > MWS** page.
2. For every source and target My webMethods Server, click **Configure MWS Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the My webMethods Server. For information about selecting the version, see "Connecting to webMethods Servers" on page 54 .
Root folder aliases	My webMethods Server aliases to use as root folders when selecting pages to deploy. Separate the folders using commas.
Include security dependencies	Whether to include the following in the dependencies list for My webMethods Server assets when creating an MWS deployment set: <ul style="list-style-type: none"> ■ Security realms that contain the assets. ■ User/group/role references in the assets' security ACLs. <p>If the dependencies do not exist on the target My webMethods Servers, include them in the list. You will have to include them in the deployment set. Exclude the dependencies if they do exist on the target My webMethods Servers. For information about resolving dependencies, see "Resolving Dependencies" on page 113 (for runtime-</p>

Field	Entry
	based deployment), or "Resolving Dependencies" on page 120 (for repository-based deployment).
Maximum Folder Object Count	Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set.
Maximum Folder Depth	Maximum number of My webMethods Server folder levels to display when you are defining and choosing assets to include in an MWS deployment set.
Enable additional MWS logging	Whether to log debug information about selected assets to source My webMethods Server logs, and assets that Deployer deploys to target My webMethods Server logs.
Exclude Core Task Engine Dependencies	Whether to exclude Task Engine portlets from the dependencies list for task application assets. Exclude the portlets from the list if the target My webMethods Servers host the Task Engine; the portlets are installed with the Task Engine. Include the portlets if the target My webMethods Servers do not host the Task Engine; you will have to include the portlets in the deployment set. For information about dependencies, see "Resolving Dependencies" on page 113 (for runtime-based deployment) or "Resolving Dependencies" on page 120 (for repository-based deployment).
Cache Timeout	Length of time queries should remain in the cache unless the cache capacity is exceeded.
Use SSL	Whether Deployer should use SSL to connect to the My webMethods Server.
	<p>Note: You can only use SSL if the My webMethods Server is configured to use SSL. Configure the My webMethods Server's HTTPS port to <i>not</i> request client certificates. For instructions on defining the HTTPS port, see <i>Administering My webMethods Server</i>.</p>

- Click **Configure**. To test the connection, click .

Connecting to Event Servers

Use the following procedure to set up connections to target Event Servers. You cannot configure connections to source Event Servers.

Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

Note: You can set up connections to target Event Servers only when using repository-based deployment.

To connect to target Event Servers

1. In Deployer, go to the **Servers > Event Server** page.
2. Click **Configure Event Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Event Server. For information about selecting the version, see " Connecting to webMethods Servers " on page 54.
Use SSL	Whether Deployer should use SSL to connect to the Event Server.

Note: You can only use SSL if the Event Server is configured to use SSL. Configure the server's HTTPS port to *not* request client certificates. For instructions

Field	Entry
	on defining the HTTPS port, see <i>webMethods Integration Server Administrator's Guide</i> .

- Click **Configure**. To test the connection, click .

Connect to Broker Servers

For each source and target Broker, Deployer must connect to the Broker Server that controls that Broker.

To connect to source and target Broker Servers

- In Deployer, go to the **Servers > Broker** page.
- For every source or target Broker, click **Configure Broker Server** and complete these fields:

Field	Entry
Name	Name to assign to the Broker Server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = < ' ' "
Host	Host name or IP address of the Broker Server.
Port	Port for the Broker Server.
Version	Version of the Broker Server that matches the version of the project as defined by the host server alias. For information about selecting the version, see " Connecting to webMethods Servers " on page 54.
Broker Name	Name of the source or target webMethods Broker.
Client Group	Client group for Deployer to use to access the source or target Broker Server. For target Broker Servers, type <code>admin</code> . Note: To connect, Deployer must belong to the specified Broker client group and have access permission.
Context	JNDI context to use when the Broker Server serves as a JNDI provider.

Field	Entry
Client Authentication	<p>Whether Deployer should use client authentication to connect to the Broker Server. Select:</p> <ul style="list-style-type: none"> ■ None to connect to the Broker Server without any client authentication. ■ SSL to connect to the Broker Server using SSL authentication. If you select this option, you must complete the Deployer Keystore, Keystore Type, Keystore Password, DeployerTruststore, and Truststore Type fields. <p>Note: You can only use SSL if the Broker Server is configured to use SSL authentication.</p> <ul style="list-style-type: none"> ■ Basic Authentication to connect to the Broker Server using basic authentication. If you select this option, you must complete the Username and Password fields. <p>Note: You can only use Basic Authentication if the Broker Server is configured to use basic authentication.</p>
Deployer Keystore	<p>Full path to Deployer's keystore file. The keystore contains the SSL credentials (private key and signed certificate) that the Broker Server uses to authenticate Deployer's identity and establish an SSL connection. Required if you specify SSL for Client Authentication.</p>
Keystore Type	<p>File type of Deployer's keystore file. Required if you specify SSL for Client Authentication.</p>
Keystore Password	<p>Password that Deployer uses to access its keystore file. Required if you specify SSL for Client Authentication.</p>
DeployerTruststore	<p>Full path to Deployer's truststore file. The truststore contains the trusted roots for Deployer's SSL certificates. Required if you specify SSL for Client Authentication.</p>
Truststore Type	<p>File type of Deployer's truststore file. Required if you specify SSL for Client Authentication.</p>
Username	<p>Basic authentication user name. Required for Basic Authentication.</p>

Field	Entry
Password	Basic authentication password. Required for Basic Authentication .

- Click **Configure**. To test the connection, click .

Connect to BPM Process Model Servers

A process model server is an Integration Server that hosts the webMethods Process Engine and executes business processes.

To connect to source and target BPM (ProcessModel) servers

- Make sure the Process Audit Log database component is installed and Integration Server is configured to write to it. For instructions, see *Installing Software AG Products*.
- In Deployer, go to the **Servers > BPM(ProcessModel)** page.
- For every source or target ProcessModel server, click **Configure BPM(ProcessModel) Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the server. For information about selecting the version, see " Connecting to webMethods Servers " on page 54.
Use SSL	Whether Deployer should use SSL to connect to the server.

4. Click **Configure**. To test the connection, click .
5. In Designer, a logical-to-physical server mapping is defined for each ProcessModel server. For deployment purposes, you must duplicate the mapping for each process model to deploy on the model's source and target ProcessModel servers. In the Integration Server Administrator for each of the servers, do the following:
 - a. Define the physical servers in the mapping as remote servers. For instructions, see *webMethods Integration Server Administrator's Guide*.

Note: Each Integration Server hosting a process model in a cluster must be configured to use the same alias name and port number as the remote alias defined for the cluster. For more information about deploying to clustered Integration Servers, see "[Deploying to Clustered Integration Servers](#)" on page 245.

- b. Go to the **Packages > Management** page and click  for the WmDesigner package.
- c. Click **Add Logical Server** and complete these fields:

Field	Entry
Name	Name of a logical server in the mapping for the ProcessModel server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
	Note: In clusters, both the source and target ProcessModel servers must use the same logical server name.
Physical Server	Physical server to which the logical server is mapped. This should be the same value as the remote server alias name you defined in step 5a.

- d. Click **Add Logical Server**.
 - e. Repeat these steps to duplicate the rest of the mapping.
 - f. Repeat these steps for every process model you want to deploy.
6. Install the WmDeployerResource package on each ProcessModel server that will run process steps. In Deployer, go to the **Servers > IS & TN** page; the page lists all ProcessModel servers you defined as remote servers. In the **Install** column, select the check box next to each ProcessModel server and click **Install**.
 7. If a process model to deploy includes a task, go to the **Packages > Management** page on the model's source and target ProcessModel servers, click  for the WmTaskClient package, and identify the My webMethods Server that hosts the task.

Connect to Universal Messaging Servers

Use the following procedure to set up connections to target Universal Messaging servers.

Note: You can set up connections to target Universal Messaging servers only when using repository-based deployment.

To connect to target Universal Messaging servers

1. In Deployer, go to the **Servers > UniversalMessaging** page.
2. Click **Configure UniversalMessaging Server**. In the **Configure UniversalMessaging Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Realm URL	The URL of the Universal Messaging server in the following format: <i>protocol</i> :// <i>host</i> : <i>port</i> Where: <ul style="list-style-type: none"> ■ <i>protocol</i> is one of the following: <ul style="list-style-type: none"> ■ <i>nsp</i> specifies Universal Messaging Socket Protocol. ■ <i>nhp</i> specifies Universal Messaging HTTP Protocol. ■ <i>nsps</i> specifies Universal Messaging Socket Protocol Secure (using SSL/TLS). ■ <i>nhps</i> specifies Universal Messaging HTTP Protocol Secure (using SSL/TLS). <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p>Note: If you use <i>nsps</i> or <i>nhps</i>, you must specify values for Deployer Keystore, Keystore Password, Deployer Truststore and Truststore Password.</p> </div> <ul style="list-style-type: none"> ■ <i>host</i> specifies the host on which the server is running. ■ <i>port</i> specifies the port number of the server.

Field	Entry
Version	Version of the Universal Messaging server. For information about selecting the version, see " Connecting to webMethods Servers " on page 54.
Client Authentication	<p>Whether Deployer should use client authentication to connect to the Universal Messaging server. Select:</p> <ul style="list-style-type: none"> ■ None to connect to the Universal Messaging server without any client authentication. ■ SSL to connect to the Universal Messaging server using SSL authentication. If you select this option, you must complete the Deployer Keystore Keystore Password, Deployer Truststore, and Truststore Password fields. <p>Note: You can only use SSL if the Universal Messaging server is configured to use SSL authentication.</p> <ul style="list-style-type: none"> ■ Basic Authentication to connect to the Universal Messaging server using basic authentication. If you select this option, you must provide information in the Username and Password fields. <p>Note: You can only use Basic Authentication if the Universal Messaging server is configured to use basic authentication.</p>
Deployer Keystore	Full path to the keystore file. Required if the protocol of Realm URL is nsps or nhps.
Keystore Password	Password used to access the keystore file. Required if the protocol of Realm URL is nsps or nhps.
Deployer Truststore	Full path to the truststore file. Required if the protocol of Realm URL is nsps or nhps.
Truststore Password	Password used to access its truststore file. Required if the protocol of Realm URL is nsps or nhps.
Username	Basic authentication user name. Required for Basic Authentication .
Password	Basic authentication password. Required for Basic Authentication .

- Click **Configure**. To test the connection, click .

Connecting to EDA Deployment Endpoints

Every Software AG installation contains suite-wide EDA assets (event type schema definitions). These assets are common for all OSGi-enabled products running within a particular installation. The Software AG Platform Manager runtime exposes an EDA deployment endpoint that enables users to deploy EDA assets.

Use the following procedure to set up connections to EDA deployment endpoints.

Note: You can set up connections to target EDA deployment endpoints only when using repository-based deployment.

To connect to target EDA deployment endpoints

- In Deployer, go to the **Servers > EDA** page.
- For every target EDA endpoint, click **Configure EDA Server**. In the **Configure EDA Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the EDA deployment endpoint. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Software AG Platform Manager runtime.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Software AG Platform Manager runtime. For information about selecting the version, see " Connecting to My webMethods Servers " on page 59.
Use SSL	Whether Deployer should use SSL to connect to the server. Configure the server's HTTPS port to not request client certificates.

- Click **Configure**. To test the connection, click .

Connecting to Business Rules Integration Servers

Use the following procedure to set up connections to target Business Rules Integration Servers.

Note: You can set up connections to target Business Rules servers only when using repository-based deployment.

To connect to target Business Rules Integration Servers

- In Deployer, go to the **Servers > Business Rules IS** page.
- For every source and target business rule server, click **Configure Business Rules IS Server**. In the **Configure Business Rules IS Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the runtime. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Integration Server running business rules.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the business rules runtime. For information about selecting the version, see "Connecting to webMethods Servers" on page 54 .
Use SSL	Whether Deployer should use SSL to connect to the server. Configure the server's HTTPS port to not request client certificates. For instructions on defining the HTTPS port, see <i>webMethods Integration Server Administrator's Guide</i> .

- Click **Configure**. To test the connection, click .

Connecting to Business Rules My webMethods Servers

Use the following procedure to set up connections to target Business Rules My webMethods Servers.

Note: You can set up connections to target Business Rules servers only when using repository-based deployment.

To connect to target Business Rules My webMethods Servers

1. In Deployer, go to the **Servers > Business Rules MWS** page.
2. For every source and target business rule server, click **Configure Business Rules MWS Server**. In the **Configure Business Rules MWS Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the runtime. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * ; , + = > < ' ' "
Host	Host name or IP address of the Integration Server running business rules.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the business rules runtime. For information about selecting the version, see "Connecting to webMethods Servers" on page 54 .
Root Context	The URL that you use to access My webMethods Server includes the server host name and port number (http://host_name:port_number). However, if necessary, you can also specify a root context path (http://host_name:port_number/root_context). For example, http://localhost:8585/mws.
Use SSL	Whether Deployer should use SSL to connect to the server. Configure the server's HTTPS port to not request

Field	Entry
	client certificates. For instructions on defining the HTTPS port, see <i>webMethods Integration Server Administrator's Guide</i> .

- Click **Configure**. To test the connection, click .

Connecting to a Repository for Repository-Based Deployment

For repository-based deployment, you define the repository as the source server. This location identifies the repository directory from which the assets should be deployed. Perform the following tasks to define a source repository for repository-based deployment.

To define the source repository

- In Deployer, go to the **Deployer > Repository** page.
- Click **Add Repository**.

Deployer displays the repository properties in the right-hand pane.

- In the **Name** field, type the name to use for the repository alias. The name can be up to 32 characters long and cannot contain spaces or the following special characters:

`$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "`

- In the **File Directory** field, type the full path of the repository directory in which the composites are located.

Note: The repository directory must be accessible to Deployer.

- Click **Configure**.

Deployer displays the repository in the table on in the **Repositories** pane.

- To test the connection, click  in the **Test** column of the **Repositories** pane.
- To rebuild the index, see ["Rebuilding the Index" on page 46](#).

Editing Properties for Source and Target Servers

Use the following procedure to edit the properties for webMethods source and target servers.

Note: For Integration Server and Trading Networks remote server aliases, you can edit only the **Version** property using Deployer. You can edit all other

properties for Integration Server and Trading Networks remote server aliases using Integration Server Administrator. For more information about editing remote server aliases, see *webMethods Integration Server Administrator's Guide*. For information about refreshing Integration Server and Trading Networks remote server aliases in Deployer, see step 4 in "[Connecting to Integration Servers and Trading Networks Servers](#)" on page 56.

Editing source and target server properties

1. In Deployer, go to the **Servers** > *server* page.
2. To edit the version of the server, select the version of the server hosting the source or target from the **Version** box and click **Save**. You can edit the version for more than one server at a time.

For information about selecting the version of the server, see "[Connecting to webMethods Servers](#)" on page 54.

3. To edit additional server properties, click the name of the server to edit. Deployer displays the server properties in the right-hand pane.
4. Edit the fields for the server as necessary.
5. Click **Save**.

Creating Target Groups

If you find that you repeatedly have to map deployment sets to the same set of target servers, you can reduce your effort by grouping the target servers into a *target group*. You can then map the deployment sets to the target group rather than to the individual target servers.

Keep in mind that not all runtime types support the use of target groups when deploying to a clustered environment. The following table describes which runtimes do and do not support the use of target groups when deploying to a clustered environment:

Runtime	Use target groups when deploying to a clustered environment?
Application Platform	Yes. To deploy Application Platform assets to a clustered environment, you must set up connections to the cluster and create a target group that includes all of the servers in the cluster.
BAM process models	No.
BPM process models	Yes. For instructions on creating target groups for use in a clustered environment, see " Deploying to Clustered Integration Servers " on page 245.

Runtime	Use target groups when deploying to a clustered environment?
Broker	Yes. To copy Broker clients to all of the cluster nodes, you must use a target group to deploy the Broker clients to each node in the cluster. When you deploy Broker assets to one of the nodes in a Broker cluster, all Broker assets except clients are copied to all of the cluster nodes.
Business rules	Yes. To deploy business rules to a clustered environment, you should set up connections to the cluster and create a target group that includes all of the servers in the cluster.
EDA	Yes. To deploy EDA assets to a clustered environment, you must set up connections to the cluster and create a target group that includes all of the servers in the cluster.
Event Servers	Yes. You <i>must</i> use target groups to deploy to event servers in an HA cluster. You can also use a target group when deploying to multiple independent event servers. Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.
Integration Server	Yes. For instructions on creating target groups for use in a clustered environment, see " Deploying to Clustered Integration Servers " on page 245.
My webMethods	Yes. To deploy My webMethods assets to a clustered environment, you should set up connections to the cluster and create a target group that includes all of the servers in the cluster.
Optimize	No. When deploying Optimize assets to an Optimize cluster, you should deploy to a single node of that cluster. Do not deploy to a target group.
Trading Networks	Yes. For instructions on creating target groups for use in a clustered environment, see " Deploying to Clustered Integration Servers " on page 245.
Universal Messaging	Yes. You <i>must</i> use target groups to deploy cluster wide assets. For all other assets, you can use target groups if you want to sync the asset properties on each node.

To create a target group

1. In Deployer, go to the **Target Groups** > *server* page.
2. Click **Create server Groups**.
3. In the **Name** field, type the name to use for the target group. The name can be up to 32 characters long and cannot contain spaces or the following special characters:
\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
4. In the **Description** field, type a description for the target group. The description length has no limit and can include any characters.
5. In the **Version** box, enter the version of the target group.

Note: Deployer limits the servers you can select for inclusion in the target group to those with the version you specify in the **Version** box. You cannot add servers of different versions to a target group. For information about selecting the version, see ["Connecting to webMethods Servers" on page 54](#).

6. Click **Create**.
7. You can specify that deployment must either succeed on all servers in the target group or be automatically rolled back. In other words, if deployment fails on any server in the target group, you can specify that Deployer must automatically roll back the deployment on all servers in the group. To do so, set **Rollback All on Failure** to **Yes**; Deployer will ignore the **Rollback on Error** project setting (see ["Setting General Deployment Defaults" on page 79](#)).

Note: **Rollback All on Failure** is valid for runtime-based deployment only. Deployer ignores this setting for repository-based deployment.

8. The **Available Servers** list shows the servers of the specified type for which you have set up connections to Deployer and that match the version you specified in the **Version** field. Select the servers to add to the target group, and then click **Add**. The servers move to the **Selected Servers** list.
9. Click **Save**.
10. To test the connection between Deployer and the target group, click  in the **Test** column in the left pane.

If the test fails, Deployer displays  **Resolve** in the **Test** column. You must resolve the servers to continue. Perform the following task to resolve the servers within the target group:

- a. Click  **Resolve** in the **Test** column.

Deployer displays the unresolved servers in the Check Inconsistencies page in the right pane.

- b. Click the server to resolve and click **Resolve Inconsistencies**.
Deployer removes the server from the target group and returns you to the Configure Target Group page.
- c. Make any additional changes to the target group and click **Save**.
- d. Click  in the **Test** column in the left pane to test the connection between Deployer and the target group.

11. If you want to rebuild the index, perform the following procedure:

Note: For more information about rebuilding the index, see "[Rebuilding the Index](#)" on page 46.

- a. If you want to change the repository directory, perform the following:
 - a. In the **Name** column, click the name of the repository to edit.
Deployer opens the repository properties in the right-hand pane.
 - b. In the **File Directory** box, type the full path of the repository for which to rebuild the index.
 - c. Click **Save Changes**.
- b. In the **Create Index** column for the repository, click .

For more information about rebuilding the index, see "[Rebuilding the Index](#)" on page 46.

click  in the **Create Index** column on the left pane.

Next Steps

After you have connected to the source and target servers, you can create a project. For more information on creating a project, see "[Creating and Managing Projects](#)" on page 77.

5 Creating and Managing Projects

■ Enabling or Disabling Deployer GUI Audit Logging	78
■ Setting Default Properties for All Projects	79
■ Setting the Defaults for Integration Server and Trading Networks Projects	82
■ Creating a Project	85
■ Exporting and Importing Project Properties	89
■ Permissions for Performing Tasks in Projects	90
■ Adding and Viewing Instructions or Notes About a Project	92
■ Editing Settings for an Individual Project	93
■ Deleting a Project	93
■ Next Steps	93

Enabling or Disabling Deployer GUI Audit Logging

Under the **Audit Logging Settings** area on the **Settings** page, specify whether to enable or disable audit logging for user actions taken through the Deployer GUI.

To view the audit log, from the Integration Server Administrator go to the **Logs > Audit** page. The columns in the audit log are described below.

Column	Description						
Time Stamp	Date and time the entry was written to the log.						
Request Type	Type of action Deployer performed (for example, Create , Build , or Deploy).						
Message	Message that describes the action.						
Status	Outcome of the action (for example, Success or Failed).						
User Id	User name under which Deployer performed the action.						
Server	Details about the server. See Type , Alias , and Host IP:Port .						
	<table border="1"> <tr> <td>Type</td> <td>Type of server on which the action was performed; can include the Integration Server that hosts Deployer and source and target servers.</td> </tr> <tr> <td>Alias</td> <td>Name assigned to the server in Deployer (see "Starting Deployer and Connecting to Servers" on page 53).</td> </tr> <tr> <td>Host IP:Port</td> <td>IP address and port for the server on which the action was performed.</td> </tr> </table>	Type	Type of server on which the action was performed; can include the Integration Server that hosts Deployer and source and target servers.	Alias	Name assigned to the server in Deployer (see "Starting Deployer and Connecting to Servers" on page 53).	Host IP:Port	IP address and port for the server on which the action was performed.
Type	Type of server on which the action was performed; can include the Integration Server that hosts Deployer and source and target servers.						
Alias	Name assigned to the server in Deployer (see "Starting Deployer and Connecting to Servers" on page 53).						
Host IP:Port	IP address and port for the server on which the action was performed.						
Thread ID	Unique identification for each Deployer action. If an action fails, the thread ID helps you identify the failed action through the Audit Log page in Deployer. Using this, the user can identify the failed activity.						

To change this display, use the **Log display controls** area at the top of the page and then click **Refresh**. The changes remain until you change them again, or until you shut down the Integration Server that hosts Deployer, whichever comes first.

Setting Default Properties for All Projects

Deployer uses default properties for all projects. To set the properties, go to the **Deployer > Settings** page. When you are done, click **Save**.

You can override many of these properties for individual projects. For instructions, see ["Creating a Project" on page 85](#).

Setting the Dependency Checking Default

Under the **Dependency Checking Default** area on the **Settings** page, indicate the default for dependency checking for all projects.

Note: Dependency checking settings apply only to runtime-based deployment.

Option	Description
Always	Tells Deployer to automatically check dependencies regularly as you modify the project and progress through the different phases of deployment.
Reduced	Tells Deployer to automatically check dependencies when you create a project build and when you deploy. You can trigger additional dependency checking at different steps yourself.
Manual	You will trigger dependency checking at different steps yourself.

Setting the Project Locking Default

Deployer offers project locking. If locking is enabled for a project, a user who wants to modify the project or perform an action such as deployment must lock the project. Other users will be able to view the project and run display commands. If necessary, users with administrative privileges can unlock a locked project.

To set project locking

Under the **Project Locking Default** area on the **Settings** page, indicate whether locking should be enabled or disabled by default for all projects.

Setting General Deployment Defaults

Under the **General Deployment Defaults** area on the **Settings** page, indicate the defaults for the following options for all projects.

Option	Entry
Large File Support	<p>For Integration Server packages and webMethods files, indicate whether you want Deployer to stream from the source server to Deployer and from Deployer to the target server by default for all projects.</p> <ul style="list-style-type: none"> ■ If you choose not to stream, the build size of a project containing Integration Server packages and webMethods files cannot exceed the amount of RAM configured for the source or target Integration Server or Deployer. If it does, you will have to distribute the Integration Server packages and webMethods files across multiple projects (and thus multiple project builds) instead. ■ If you choose to stream, the build size of a project containing Integration Server packages and webMethods files can be up to 4GB. For streaming to work, you must set certain extended settings on every source and target Integration Server, and on the Integration Server that hosts Deployer. <p>In Integration Server Administrator, go to the Settings > Extended > Edit Extended Settings page. Type the following server configuration parameters and values in the box. For complete information about these server configuration parameters, see the <i>webMethods Integration Server Administrator's Guide</i>.</p> <ul style="list-style-type: none"> ■ <code>watt.server.tspace.timeToLive=time</code> ■ <code>watt.server.tspace.location=full_path_on_local_machine</code> ■ <code>watt.server.tspace.max=maximum</code>
	<p>If your project contains BPM process models or Integration Server packages, set the following additional server configuration parameters:</p> <ul style="list-style-type: none"> ■ <code>watt.server.SOAP.MTOMStreaming.enable=true</code> ■ <code>watt.server.SOAP.MTOMStreaming.cachedFiles.location=directory_path</code> ■ <code>watt.server.SOAP.MTOMStreaming.threshold=number_of_bytes</code>
	<p>Click Save Changes and restart the Integration Server.</p>
	<p>Note: Streaming is available only over HTTP. If your source or target environment uses HTTPS, you cannot choose to stream. This option is available for runtime-based deployment only.</p>
Optimize UI Response	<p>(Runtime-based deployment only) Click Yes to increase the Deployer GUI responsiveness for large projects. This setting causes Deployer to bypass the ping requests from the GUI that Deployer usually</p>

Option	Entry
	performs during the map and define dependency checks. The default setting is No .
Checkpoint Creation	(Runtime-based deployment only) To have Deployer automatically create a checkpoint for target servers before deploying, click Automatic . If you want to generate checkpoints when you choose, click Manual .
	<p>Note: If deployment sets in a project include deletion sets, Deployer ignores these settings. Instead, Deployer automatically creates checkpoints for target servers.</p>
Rollback on Error	(Runtime-based deployment only) To have Deployer automatically create checkpoints before deploying and roll back deployments if they fail, click Automatic . If you want to roll back deployments when you choose, click Manual .
	<p>Note: If deployment sets in a project include deletion sets, Deployer ignores these settings. Instead, Deployer automatically rolls back the target servers.</p>
Enable Concurrent Deployment	To set Deployer to deploy assets concurrently, click Yes (the default). If you want to deploy assets sequentially, click No . For more information about concurrent and sequential deployment, see " Concurrent and Sequential Deployment " on page 21.
Deployer Service Timeout	<p>The amount of time (in milliseconds) to wait before Deployer services time out waiting for a response.</p> <p>This setting overrides the <code>watt.net.timeout</code> and <code>watt.server.SOAP.request.timeout</code> server configuration properties for Deployer.</p>
	<p>Note: When using the Integration Server or BPM server ping operations, Deployer calls a service that returns cluster information. If the target Integration Server or BPM server is down, this service can take a long time to time out. To reduce the amount of time to wait, set the <code>clusterServiceInvokeTimeout</code> property in the <code>WmDeployer/config/deployer.cnf</code> file. The default value is 2000 milliseconds. Change this value according to your system's requirements and reload the Deployer package for the changes to take effect.</p>
Batch Size for	Sets the number of assets that Deployer deploys at one time for runtime-based deployment. Enter the maximum number of assets

Option	Entry
Runtime Deployment	<p>to build and deploy as a batch at one time. This limits the number of assets in each deployment to only the number indicated. The default value is 10.</p> <p>By batching deployments, you break the transportation of large assets into smaller batches, which decreases the memory utilization.</p> <p>If you set this property to 0, then Deployer does not limit the number of assets during build, checkpoint, rollback, and deploy operations.</p>
Batch Size for Repository Deployment	<p>Sets the number of assets that Deployer deploys at one time for repository-based deployment. Enter the maximum number of assets to build and deploy as a batch at one time. This limits the number of assets in each deployment to only the number indicated. The default value is 1.</p> <p>Note: If you are deploying BPM or Optimize assets, set this parameter to 0.</p>
Maximum Plugin Objects Displayed	<p>Sets the maximum number of assets Deployer displays in the tree for deployment and deletion sets. The default is 2500.</p>

Setting the Defaults for Integration Server and Trading Networks Projects

Note: **IS & TN Deployment Defaults** settings apply only to runtime-based deployment.

In addition to the default properties for all projects, Deployer uses default properties for all Integration Server and Trading Networks projects. To set these properties, go to the **Deployer > Settings** page. When you are done, click **Save**.

You can override many of these properties for individual projects; see "[Creating a Project](#)" on page 85.

These default properties apply to all assets except Integration Server packages. You specify package properties for Integration Server packages on a package-by-package basis. For instructions, see "[Setting Package Properties](#)" on page 108.

To set default properties for Integration Server and Trading Networks projects

1. In the **Suspend During Deployment** area on the **Settings** page, indicate whether Deployer should suspend activity for the Integration Server assets listed below while

deployment is going on. Typically, if the targets are production Integration Servers, you would suspend all of these types of assets. After deployment, Deployer enables the disabled ports and resumes the suspended triggers, adapter listeners, polling notifications, and scheduled tasks.

Asset	Description	Click...
Triggers	Allow all running trigger operations to complete, then suspend all trigger execution and document retrieval on the target Integration Servers.	All
	<p>Note: If you choose All, Deployer suspends execution and document retrieval for ALL triggers on the target Integration Servers, not just for the triggers that you include in the project.</p>	
	Do not suspend triggers.	None
	Suspend individual triggers. You choose the triggers to suspend when you set package properties (see " Setting Package Properties " on page 108).	Selected
Ports	Whether to disable ports on the target Integration Servers that match ports you are trying to deploy.	
Scheduled Tasks	Whether to prevent scheduled tasks on the target Integration Servers that match scheduled tasks you are trying to deploy from running.	
	<p>Note: Tasks that are already running at deployment time are not affected by deployment.</p>	
Adapters	Do not suspend adapter listeners or polling notifications.	None
	Suspend individual adapter listeners and polling notifications. You choose the notifications to suspend when you set package properties (see " Setting Package Properties " on page 108).	Selected

- In the **Overwrite Existing** area on the **Settings** page, indicate how Deployer should proceed when it finds that assets you are trying to deploy already exist on target Integration Servers.

For this option...	Indicate whether Deployer should...	Click
TN Rules	Replace the entire rule list.	Replace All
	Overwrite existing rules and deploy new rules into the rule set.	Merge
ACL Maps	Deploy the mapping of ACLs to services for any services you choose to deploy. Deploy ACL maps if you want to assign the same ACLs to the deployed services on the target Integration Server that you assigned to the source services on the source Integration Servers.	
Other Non-Package Assets	Overwrite existing assets. This option applies to all assets except the following: <ul style="list-style-type: none"> ■ Trading Networks processing rules (see the previous step). ■ Integration Server ACL maps (see ACL Maps, above). ■ Integration Server packages. You specify the overwrite option for Integration Server packages on a package-by-package basis, as described in "Setting Package Properties" on page 108. 	
Scheduled Tasks By	This property can be set to either Service Name or ID . If this property is set to Service Name , and if a scheduled task with the same service name already exists, Deployer overwrites it with the one being deployed. If this property is set to ID , then based on the task ID which is present on the target, Deployer either overwrites the task if the source and target task IDs are same, or creates a new one.	
Note:	Before you deploy a project for runtime-based deployment, you can find out which assets Deployer will overwrite by generating the simulation report.	

- In the **Activate After Deployment** area on the **Settings** page, indicate whether Deployer should activate the assets below on the target Integration Servers. **Activate After Deployment** is used *only* if **Suspend During Deployment** is set to **Yes**.

For this option...	Indicate whether Deployer should...
Ports	Activate newly deployed ports. Note: If you choose to activate ports, and one of the ports you deploy uses the same port number as an existing port on a target Integration Server, Deployer will display a message to that effect and will not activate the port.
Scheduled Tasks	Activate newly deployed scheduled tasks.
Adapters	Activate newly deployed adapter connections, notifications, and listeners. Note: If you choose to not activate, the deployed adapter connections, notifications, and listeners will be in the same state they are in on the source server.

- In the **Packages** area on the **Settings** page, indicate whether Deployer should compile the package during deployment on the target Integration Server.

For this option...	Indicate whether Deployer should...
Compile Java Services	Compile the package during deployment. When Integration Server packages contain Java sources and are deployed using runtime-based deployment, the Java sources will get compiled on the target Integration Server if the Compile Java Services property is set to Yes .

- Click **Save**.

Creating a Project

You can create a project by creating a new, blank project or by copying an existing project and modifying it.

To create a project

- Go to the **Deployer > Projects** page.
- Create a project using one of these methods:

- To create a new project:
 - i. Click **Create Project**.
 - ii. In the **Name** box, type the name to use for the new project. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:
 $\$ \sim / \backslash \# \& @ \wedge ! \% * ; , + = > < ' ' "$
 - iii. In the **Description** box, type a description for the project. The description length has no limit and can include any characters.
 - iv. In the **Project Type** area, select one of the following:

To create a...	Click...
Runtime-based deployment project	Runtime
Repository-based deployment project	Repository

- v. Click **Create**.
- To create a project from an existing project:
 - i. Click **Copy Project**.
 - ii. In the **Project to Copy** box, click the project to copy.
 - iii. In the **New Project Name** box, type the name to use for the new project. The name can be up to 32 characters long and can include any characters that are valid for a file name in your operating system.
 - iv. Click **Copy Project**.
3. Review the default properties for projects in the right-hand pane and override for the project you are creating if necessary.

In the right-hand pane, Deployer displays those properties for which you can override the settings for an individual project. By default, these properties inherit the values set for the default as described in "[Setting Default Properties for All Projects](#)" on page 79. For example, if the global property for the **Enable Concurrent Deployment** property is set to **Yes** (for concurrent deployment), you can set an individual project to use sequential deployment by setting this property to **No**.

4. Depending on the specific type of project you are creating, you can set the following additional properties in the right-hand pane:

Note: For an explanation of the project properties you can set for all deployment projects, see "[Setting Default Properties for All Projects](#)" on page 79.

- If you are creating a repository-based project, you can set the following additional properties:

<u>For this property...</u>	<u>Indicate whether Deployer should...</u>
Enable Project Locking	Whether locking is enabled or disabled for the project. Click Yes to enable locking. Click No to disable locking.
Enable Concurrent Deployment	Deploy assets concurrently. Click Yes to deploy assets concurrently. If you want to deploy assets sequentially, click No . For more information about concurrent and sequential deployment, see " Concurrent and Sequential Deployment " on page 21.
Ignore Missing Dependencies	Ignore missing dependencies. If you click Yes , Deployer deploys the composite even when the dependent composite is not available on the repository.
Enable Transactional Deployment	<p>Automatically create a checkpoint prior to delivering and activating deployment and deletion sets. If set to Yes (the default), transactional deployment is enabled.</p> <p>When transactional deployment is enabled and activation fails, Deployer triggers a roll back automatically and restores the target servers to the state of the prior activation.</p>

- If the project is for IS & TN, see "[Setting the Defaults for Integration Server and Trading Networks Projects](#)" on page 82.
- If the project is for Optimize, you can set the following properties under the **Optimize Options** area:

Note: The following properties are available only for Deployer 8.2 SP1 and earlier.

<u>For this property...</u>	<u>Indicate whether Deployer should...</u>
Include Dimension Values	Indicates whether Deployer should include the values for dimensions you add to deployment sets (for example, Customer Names or Product Types).

<u>For this property...</u>	<u>Indicate whether Deployer should...</u>
Display Data Definition Statements	Indicates whether Deployer should display the values for data definition statements in the binary stored in the project build.

- If the project is for process models, you can set the properties below for the project under the **ProcessModel Deployment Options** area. For more information about process models, see *webMethods Monitor User's Guide*.

<u>For this property...</u>	<u>Indicate whether Deployer should...</u>
Enable process for execution	<p>Enable webMethods-executed business process versions for execution after deployment. When a process version is enabled, the Process Engine uses the enabled version when starting new process instances. When a process is disabled, the Process Engine does not use the process version for new process instances.</p> <p>Only one version of a process can be enabled at a time. If there are no enabled process versions, the Process Engine will not start any process instances of the process.</p>
Enable process for analysis	<p>Enable webMethods-executed processes for analysis after deployment. When a process is enabled, the Process Engine forwards all process instance activity to the Optimize Analytic Engines. When a process is disabled, no activity is forwarded.</p>

- If the project is for My webMethods Server, you can set these properties for the project under the **MWS Deployment Options** area:

<u>For this property...</u>	<u>Indicate whether Deployer should...</u>
Export Subscriptions	Deploy subscriptions for My webMethods Server assets you are deploying.
Export Access Control Lists	Deploy ACLs for My webMethods Server assets you are deploying.

<u>For this property...</u>	<u>Indicate whether Deployer should...</u>
Export Principal Attributes	Include attributes contained in attributes providers when exporting users, groups, and roles.
Export Content As Reference	Export a reference to the page content without deploying the content.
Alias Prefix	Apply the specified prefix to every automatically generated My webMethods Server alias.
Export Version History	Include all versions of an asset in Portal version control. This applies to the content within a page or folder.
Auto Generate Aliases	Automatically generate an alias on the target My webMethods Server for every My webMethods Server asset that is deployed. If an asset already has one or more aliases, then the aliases are retained when the auto-generated alias is added.
Export Content (Documents)	Deploy content referenced by portal pages and folders you are deploying (for example, a PDF document that has been published on a portal page you are deploying).
Page Depth	If the value of this property is 1, all first level child pages that are under a selected parent page are deployed, even if the child pages are not selected in the deployment set. Default value is 1. If the value of this property is 0, only those child pages that are selected will be deployed. Child pages that are not selected will not be deployed.

5. Click **Save**.

Exporting and Importing Project Properties

You can export and import project properties for deployment projects.

Note: Deployer does not export and import deletion sets as a part of this procedure.

For more information about exporting and importing deletion sets, see ["Exporting and Importing Deletion Set Definitions" on page 134](#).

When you export project properties, Deployer creates a file that contains the project property settings. The file is named *project*.properties and is stored in the *Integration Server_directory*\packages\WmDeployer\replicate\outbound directory. You can then import the project property settings into another Deployer project.

To export and import the project properties

1. Export project properties as follows:
 - a. In **Deployer**, go to the **Deployer > Projects** page.
 - b. In the **Name** column, click the project from which to export.
 - c. In the right-hand pane, click **Export Project properties**. **Deployer** creates a file that contains the project property settings. The file is named *project.properties* and is stored in the *Integration Server_directory\packages\WmDeployer\replicate\outbound* directory. **Deployer** also gives you the option to save the file to your local file system.
2. Import project properties into another project as follows:
 - a. Copy the *project.properties* file to the *Integration Server_directory\packages\WmDeployer\replicate\inbound* directory on the machine that hosts the project into which to import.
 - b. In **Deployer**, go to the **Deployer > Projects** page.
 - c. In the **Name** column, click the project into which to import.
 - d. In the right-hand pane, click **Import Project properties**, then select the *project.properties* file you just copied to the inbound directory.

Permissions for Performing Tasks in Projects

You can authorize users to perform tasks by project. To do this you use tasks and authorizing groups or My webMethods Server central user management. This means that when **Deployer** users display the **Projects** page, they will see only those **Deployer** projects for which they are authorized.

You can authorize groups to perform the following tasks:

- **View.** View projects only.

Note: Users with Developer and Internal ACLs and any combination of Define, Build, Map, or Deploy authorization automatically have the View authorization.

- **Define.** Groups can define, export, and import deployment and deletion sets only.
- **Build.** Groups can build projects only.
- **Deploy.** Groups can deploy deployment or deletion sets only (that is, to actually deploy assets to or delete assets from target servers or target groups).
- **Map.** Groups can map projects to repositories, source servers, target servers, and target groups.

You grant privileges to perform tasks through Access Control Lists (ACLs). When an administrator creates ACLs, he or she identifies groups that are allowed to perform particular tasks. The following table shows the ACLs and authorizations required to perform each task:

	All Project Tasks	View Project Tasks	Define Project Tasks	Build Project Tasks	Map Project Tasks	Deploy Project Tasks
ACLs						
Developer	X	X	X	X	X	X
Internal	X	X	X	X	X	X
DeployerAdmin	X*					
Administrator	X*					
Authorizations						
View		X				
Define			X			
Build				X		
Map					X	
Deploy						X
Note:	<p>The asterisk (*) indicates that only users with either one of the following ACL combinations can perform all project tasks in Deployer:</p> <ul style="list-style-type: none"> ■ DeployerAdmin, Developer, and Internal ACLs ■ Administrator ACL <p>Users assigned to one of these ACL combinations do not require authorization to specific tasks.</p>					

You can authorize groups to perform more than one task. For example, if you want to allow Group A to map and deploy projects, you would select the Map and Deploy authorizations.

Note: The groups you create for use with Deployer can contain unique names to help define which tasks each group can perform. For example, you could create groups named `viewDeployerProjects`, `buildDeployerProjects`, `mapDeployerProjects`, `deployDeployerProjects`, and `defineDeployerProjects`. This means that when Deployer users display the **Projects** page, they will see only those Deployer projects to which they are authorized.

For more information about groups and ACLs, see *webMethods Integration Server Administrator's Guide*. For information on My webMethods Server central user management groups, see *Administering My webMethods Server*.

To authorize groups to perform tasks

1. In Deployer, go to the **Deployer > Projects** page.
2. Locate the project to which to authorize groups. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project. In the **Authorize** column for the project, click .
3. In the **Fetch Groups** field, click the type of group to authorize.

If you select **LDAP/Central user management groups**, you can narrow the list of groups that are displayed. In the **Search String** field, specify a string that is within the names of the groups you want to display; you can use an asterisk (*) as a wildcard for one or more characters (for example, `Admin*`). Click **Go** to list the specified groups.

4. In the **Select Authorization** list, click the task the group is authorized to perform.
5. The **Not Specified** box lists all groups defined in the type of group you chose. Using the arrow buttons, move each group you want to assign to the specified task into the **Allowed** box. Move each group that you do not want to assign to the specified task into the **Denied** box.
6. Click **Update**. The **Resulting users with this Authorization** lists all users that belong to the groups you assigned to the task (that is, the groups you moved into the **Allowed** box).
7. In the **Lock Status** column for the project, click  to unlock the project.

Adding and Viewing Instructions or Notes About a Project

When you create a project, Deployer automatically creates an HTML home page for the project. The HTML home page for a project is located in the *Integration Server_directory\packages\WmDeployer\pub\projects\project* directory. The file name for the home page is *project.html*. Modify the page as necessary, but do not move it from this directory or rename it.

To view the home page for the project, go to the **Deployer > Projects** page and click  in the **Home** column for the project.

Editing Settings for an Individual Project

Use the following procedure to edit the settings for an individual project.

Note: To change the project settings for *all* projects, see ["Setting Default Properties for All Projects" on page 79](#).

To edit settings for an individual project

1. In **Deployer**, go to the **Deployer > Projects** page.
2. In the **Name** column of the **Projects** area, click the name of the project for which you want to edit the properties.

Deployer displays the properties for which you can override the settings for an individual project in the right-hand pane.

3. Make changes to the properties and click **Save**.

Deleting a Project

Use the following procedure to delete a project you no longer need. When you delete a project, Deployer also deletes the deployment and deletion sets, builds, deployment maps, and deployment candidates associated with that project.

Deployer does not delete source repositories or composites used for repository-based deployment or source and target server aliases when you delete the project.

To delete a project

1. In **Deployer**, go to the **Deployer > Projects** page.
2. Click  in the **Delete** column for the project.
Deployer displays a confirmation dialog.
3. Click **OK** to confirm that you want to delete the project.

Next Steps

Once you have set logging options and other project default settings and created a project, you can do the following:

If you are creating a...

Runtime-based deployment project

You can...

- Define a deployment set. See ["Defining a Deployment Set" on page 95](#).

If you are creating a...

You can...

Repository-based deployment project

- Define a deletion set. See ["Defining a Deletion Set"](#) on page 127.

Define a deployment set. See ["Defining a Deployment Set"](#) on page 95.

6 Defining a Deployment Set

■ Creating a Deployment Set	96
■ Identify Source Servers for the Deployment Set	97
■ Next Steps	99

Creating a Deployment Set

You create a deployment set to define the assets Deployer deploys to target servers. Perform the following procedure to create a deployment set for either a runtime-based or repository-based deployment.

To create a deployment set

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Define**.
5. Click **Create Set**.
6. If you are creating a deployment set for runtime-based deployment, perform the following:
 - a. From the **Type** box, select the type of deployment set you want to create.
 - b. From the **Set** box, select **Deployment**.
7. Complete the following fields:

Box	Entry
Name	Name to use for the deployment set. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Description	Description for the deployment set. The description length has no limit and can include any characters.
Maximum TN Assets to Display	(Runtime-based deployment only) Number of Trading Networks assets Deployer should display. Depending on your browser, Deployer might not be able to display more than 1000 assets for Trading Networks. If the source server hosts more than 1000 assets, use this field with the with the All other assets field to reduce the number of assets displayed.
Packages(IS & TN	(Runtime-based deployment only) After you choose the source servers, Deployer will display all packages on the servers. You can use this field to narrow the display. Type

Box	Entry
deployment set)	<p>a regular expression that specifies the text that the package names must contain in order to be displayed.</p> <p>The following are examples of regular expressions:</p> <ul style="list-style-type: none"> ■ To narrow display to packages whose name starts with <i>string</i>, use the following: <i>string</i>.* ■ To narrow display to packages whose name ends with <i>string</i>, use the following: .*<i>string</i>\$
All other assets	<p>(Runtime-based deployment only) After you choose the source servers, Deployer will display all assets on those servers. You can use this field to narrow the display. Specify a regular expression that specifies the text that the asset names must contain in order to be displayed. For examples, see the Packages field.</p>

8. Click **Create**.

Identify Source Servers for the Deployment Set

You must define source servers from which Deployer obtains the assets for deployment.

Perform one of the following procedures to identify the source servers for the deployment set:

For...	See...
Runtime-based deployment	"Identifying Source Servers for Runtime-Based Deployment" on page 97
Repository-based deployment	"Identifying Source Repository for Repository-Based Deployment" on page 98

Identifying Source Servers for Runtime-Based Deployment

You can define the following as source servers for runtime-based deployment:

- Integration Server & Trading Networks
- Optimize

- My webMethods Server
- Broker
- BPM (ProcessModel)

Note: You cannot configure source servers for Event Server or business rules runtimes for runtime-based deployment.

Note: You cannot define the same server alias as both a source and target server in a deployment set.

Perform the following procedure to identify the source servers for runtime-based deployment.

To identify source servers for runtime-based deployment

1. On the **Deployer > Projects > project > Define** page, in the left-hand pane in the **Name** column, click the deployment set for which to identify source servers.
2. In the **Select** column of the **Select Source Servers** area, select the check box next to each source server that contains assets to add to the deployment set.

Note: Deployer lists the version of each source server in the **Version** column. You must select source servers with the same version for the deployment set. You cannot include source servers with different versions in a deployment set. For information about selecting the versions of source servers, see ["Connecting to webMethods Servers" on page 54](#).

Note: For a BPM (ProcessModel) deployment set, you can select only one source server. If you want to deploy process models from more than one ProcessModel server, you must define a deployment set for each ProcessModel server.

Note: If a server you want to use as a source does not appear in the list, you have not yet set it up to work with Deployer. For instructions, see ["Starting Deployer and Connecting to Servers" on page 53](#). Then click **Refresh this Page** to update the list of servers on this page.

3. Click **Save**.

Deployer refreshes the **Select Source Servers** list to display only source servers that share the same version as the selected source servers. To display all of the available source servers, deselect the source servers and click **Save**.

Identifying Source Repository for Repository-Based Deployment

For repository-based deployments, the source is *always* the repository on which the composites are built. Perform the following procedure to identify repository for repository-based deployment.

To identify the source repository for repository-based deployment

1. On the **Deployer > Projects > *project* > Define** page, in the left-hand pane in the **Name** column, click the deployment set from which you will select the composites to deploy.
2. In the **Select** column of the **Select Repository** area, click ***repository_alias*** that contains composites to add to the deployment set.

Note: You can select only one repository.

Note: If a repository you want to use as a source does not appear in the list, you have not yet created the alias for the repository. For instructions, see ["Connecting to a Repository for Repository-Based Deployment"](#) on page 72. Then click **Refresh this Page** to update the list of servers on this page.

3. Click **Save**.

Deployer adds the repository as a child of the deployment set in the **Name** column of the **Deployer > Projects > *project* > Define** page, in the left-hand pane.

Deployer displays the  icon for the repository.

Next Steps

Once you have defined a deployment set and the source servers or repository for your project, you can add the assets to the deployment set. You perform different tasks to add assets depending on whether you are creating a runtime-based or repository-based deployment project. See the section specific to the type of deployment project you are creating as follows:

If you are creating a...	See...
Runtime-based deployment project	See "Adding Assets for Runtime-Based Deployment" on page 101.
Repository-based deployment project	See "Adding Assets for Repository-Based Deployment" on page 117.

7 Adding Assets for Runtime-Based Deployment

■ Before Adding Assets for Runtime-Based Deployment	102
■ Adding Assets to Broker, ProcessModel, MWS, or Optimize Deployment Sets	102
■ Adding Assets to an IS & TN Deployment Set	104
■ Resolving Dependencies	113
■ Manually Adding Dependencies to a Package Component in an IS & TN Deployment Set	115
■ Removing Process Models from a Deployment Set	116
■ Next Steps	116

Before Adding Assets for Runtime-Based Deployment

Before you can add assets for runtime-based deployment, you must perform the following

1. Connect to source and target servers and optionally define target groups. For more information, see ["Starting Deployer and Connecting to Servers" on page 53](#).
2. Create a project. For more information, see ["Creating and Managing Projects" on page 77](#).
3. Define a deployment set. For more information, see ["Defining a Deployment Set" on page 95](#).

Keep the following points in mind when adding assets for runtime-based deployment:

- You can only deploy *user-created* assets using Deployer. You cannot deploy system components that were installed by the Software AG Installer as part of a product installation. For example, you can deploy Integration Server packages that were created by users, but you cannot deploy Integration Server system packages that were installed, such as the WmPRT package (Process Engine). If you want such components on target servers, you must install them using the Software AG Installer.
- You cannot define the same server alias as both a source and target server in a deployment set.

Adding Assets to Broker, ProcessModel, MWS, or Optimize Deployment Sets

You perform similar actions to add assets to Broker, Designer ProcessModels, My webMethods Server, or Optimize deployment sets.

About MWS Deployment Sets

If you are creating an MWS deployment set, the set you create should depend on the number of assets you want to deploy. The **Root folder aliases** field in the MWS Server configuration (see ["Connecting to My webMethods Servers" on page 59](#)) controls the Page assets that are displayed for each My webMethods Server. By default, this field is set to folder.public.

If your Public Folders and its subfolders contain hundreds of pages, displaying these assets can take a long time. In addition, Deployer cannot display more than 2500 assets at once, so it might not display all assets. Software AG recommends displaying smaller sets of assets and creating smaller deployment sets. To do so:

1. Change the **Root folder aliases** field to specify a folder that is deeper within the Public Folders hierarchy.

2. Create the deployment set for those assets.
3. Repeat as necessary.

About ProcessModel Deployment Sets

When deploying Designer process models with Deployer in a runtime-based project, Deployer copies the generation receipt from the source environment to the target environment. Because the logical-to-physical server mapping might contain old data, the regeneration process connects to the previous logical server and cleans up the process-related assets there (for example, deleting associated triggers and services), in addition to creating the new assets on the new logical server. With no generation receipt, this mapping information is not available and the clean-up procedure cannot occur.

Note: If you deploy multiple versions of the same process model in a single deployment set and configure the deployment set to enable both processes on the target, the version that is deployed to the target last is the one that is enabled. Use webMethods Monitor to ensure that the proper version is enabled in the target environment.

Adding Assets to a Broker, ProcessModel, MWS, or Optimize Deployment Set

Perform the following task to add assets to webMethods Broker, ProcessModel, MWS, or Optimize deployment sets.

To add assets to a webMethods Broker, ProcessModel, MWS, or Optimize deployment set

1. In the **Deployment Sets** area, under the deployment set to which to add assets, click the **Broker, ProcessModel, MWS, or Optimize** folder. In the right-hand pane, Deployer lists the source servers of the type you selected.
2. In the right-hand pane, open the tree to show the assets on the source servers, then select the check box next to each asset to add to the deployment set. Keep in mind the following:

For...	Note
ProcessModel	The process models displayed are those that were "Built for execution" on the Integration Server.
MWS	The My webMethods Server folder is listed twice within its directory, as a container preceded by  and as an asset preceded by  . If you want to add a folder with all the assets it contains to the deployment set, select the folder where it appears next to the square icon. If you want to add individual assets in the folder without adding the folder

For...	Note
	itself, open the folder where it appears as a container and click the assets to add.

3. Click **Save**. Deployer shows the selected assets in the left-hand pane under the **Broker**, **ProcessModel**, **MWS**, **Optimize**, or **Business Rules** folder for the deployment set.

Adding Assets to an IS & TN Deployment Set

The sections below explain how to add these types of user-created assets to an IS & TN deployment set:

- Integration Server administrative assets such as ports, users, groups, and scheduled tasks, packages, and web service descriptors.
- Integration Server packages.
- webMethods files.
- Trading Networks assets.

Your source and target Integration Servers might have assets in common, and you do not need to deploy them from one environment to another. You can improve Deployer performance by excluding these assets from deployment sets. For instructions, see ["Excluding Common Assets" on page 113](#).

Deploying ACLs

If you want to deploy ACLs, you must perform extra steps depending on whether the ACLs are associated with My webMethods Server groups or LDAP groups.

Deploying ACLs Associated with My webMethods Server Groups

If you want to deploy ACLs that are associated with My webMethods Server groups, you must perform the following steps.

To deploy ACLs associated with My webMethods Server groups

1. Use an MWS deployment set to deploy the My webMethods Server groups to the target Integration Servers.
2. Create an IS & TN deployment set containing the ACLs.
3. Mark the unresolved dependencies for the My webMethods Server groups as **Exists**.
4. Deploy the ACLs.

Deploying ACLs Associated with LDAP Groups

If you want to deploy ACLs that are associated with LDAP groups, perform the following steps.

To deploy ACLs associated with LDAP groups

1. Configure the LDAP groups on the target Integration Servers.
2. Create an IS & TN deployment set containing the ACLs.
3. Mark the unresolved dependencies for the LDAP groups as **Exists**.
4. Deploy the ACLs.

Adding Integration Server Administration Assets

Use the following procedure to add Integration Server administration assets.

To add Integration Server administration assets

1. In the **Deployment Sets** area, under the deployment set to which to add Integration Server administration assets, click the **Administration** folder. In the right-hand pane, Deployer lists the source Integration Servers you identified.
2. In the right-hand pane, open the tree to show the administration assets on the source Integration Servers, select the check box next to each asset to add to the deployment set, and then click **Save**. Deployer shows the selected assets in the left-hand pane under the **Administration** folder for the deployment set.
3. If you are not going to add any more assets to the deployment set, go to ["Resolving Dependencies" on page 113](#).
4. If you added JMS triggers to the deployment set, create the same JMS connection aliases on the target Integration Servers that exist on the source Integration Servers. For instructions, see *Using webMethods Integration Server to Build a Client for JMS*.

Note: If the JMS connection aliases on the target Integration Servers do not have the same names as on the source Integration Servers, the JMS triggers will not be enabled after deployment.

Adding Integration Server Packages

You can add an Integration Server package to a deployment set as follows:

- Add a package in its entirety.
- Add selected package components only (*partial* package).
- Add selected package files only (partial package).

If you add a partial package of only selected files to a deployment set and the package already exists on target Integration Servers, you can have Deployer delete specified files from the existing package on the target Integration Servers after deployment. You might use this feature if the existing package contains a service that has been superseded. In this case, you would deploy the files that make up the new service and delete the files that make up the old service.

Adding an Entire Package

To add an entire package

1. In the **Deployment Sets** area, under the deployment set to which to add packages, click the **Packages** folder. In the right-hand pane, Deployer lists all source Integration Servers.
2. In the right-hand pane, open the tree to show the packages on the source Integration Servers, select the check boxes next to the packages to add in their entirety, and then click **Save**. Deployer shows the entire package icon (📦) for the selected packages in the left-hand pane under the **Packages** folder and in the right-hand pane, and a black check mark for the packages in the right-hand pane.

Note: When you add an entire package to a deployment set, Deployer automatically also adds all ports associated with that package. Be sure to substitute configuration values for these ports if necessary (for instructions, see ["Substituting Configuration Values by Asset" on page 147](#)).

3. If you are done adding packages to the deployment set, go to ["Setting Package Properties" on page 108](#).

Adding Package Components

If you choose to add both package components and package files to a deployment set, you must be aware of the following:

- If you first select components, and then select files, Deployer only allows you to add files from the package file list.
- If you first select files, and then select components, Deployer might overwrite certain file selections to ensure consistency.

To add package components

1. In the **Deployment Sets** area, under the deployment set to which to add package components, click the **Packages** folder. In the right-hand pane, Deployer lists all source Integration Servers.
2. In the right-hand pane, open the tree to show the packages on the source Integration Servers, and then click the name of a package that contains components to add to the deployment set.

3. In the **Select Components** area, open the tree to show the components in the package, select the check box next to each component to add to the deployment set, and then click **Save**.
4. Click **Return to Packages**. Deployer shows the partial package icon (📦) for the package in the left-hand pane under the **Packages** folder and in the right-hand pane, and a gray check mark for the package in the right-hand pane.
5. If you are done adding packages to the deployment set, go to "[Setting Package Properties](#)" on page 108.

Adding Package Files

If you choose to add both package components and package files to a deployment set, you must be aware of the following:

- If you first select components, and then select files, Deployer only allows you to add files from the package file list.
- If you first select files, and then select components, Deployer might overwrite certain file selections to ensure consistency.

To add package files

1. In the **Deployment Sets** area, under the deployment set to which to add package files, click the **Packages** folder. In the right-hand pane, Deployer lists all sourceIntegration Servers.
2. In the right-hand pane, open the tree to show the packages on the sourceIntegration Servers, then click the name of a package that contains files to add to the deployment set.
3. Click **Select Files**. Deployer lists all files in the package. Do one of the following:

To add...	Do this...
All the files in the list	Click All files .
Only files you select in the list	Click Selected Files , then press the CTRL key and click each file to <i>include</i> in the deployment set. Note: The Select Files option is a link near the top of the right-hand pane.
Only files <i>other than</i> those you select in the list	Click All except selected files , then press the CTRL key and click each file to <i>exclude</i> from the deployment set.

To add...	Do this...
All files in the list whose name contains a specified string	Click Files specified by filter , then type the string on which to match the files to <i>include</i> in the deployment set. You can use an asterisk (*) as a wildcard character (for example, *.java or *.class).
All files in the list whose name does <i>not</i> contain a specified string	Click All except files specified by filter , then type the string on which to match the files to <i>exclude</i> from the deployment set. You can use an asterisk (*) as a wildcard character (for example, *.java or *.class).

- If a package of the same name as this partial package already exists on one of the deployment set's target Integration Server2s, and the existing package contains files to delete after deployment, type the fully qualified names of the files to delete in the **Files to Delete from Target** box. Type each file name on its own line, and end each line with a semicolon (;). For example:

```
code/classes/wm/administratorResource/admin.class;
code/classes/wm/administratorResource/user.class;
ns/wm/administratorResource/
```

- Click **Save**.
- Click **Return to Packages**. Deployer shows the partial package icon (📦) for the package in the left-hand pane under the **Packages** folder and in the right-hand pane, and a gray check mark for the package in the right-hand pane.
- If you are done adding packages to the deployment set, go to "[Setting Package Properties](#)" on page 108.

Setting Package Properties

You must set properties for each package you added to the deployment set.

To set properties for a package

- In the **Deployment Sets** area, under the deployment set to which you added entire or partial packages, open the tree under the **Packages** folder and click a package.
- In the **package_name Properties** area, specify the properties listed below.

Property	Entry		
Package Type	Use this property when the source package already exists on the target Integration Servers. You can use the options below for entire packages and for partial packages.		
	<table border="0"> <thead> <tr> <th><u>If you want Deployer to...</u></th> <th><u>Click...</u></th> </tr> </thead> </table>	<u>If you want Deployer to...</u>	<u>Click...</u>
<u>If you want Deployer to...</u>	<u>Click...</u>		

Property	Entry
	<p>Deploy the source package, replacing the existing package entirely. When you choose to deploy an entire package, this is the default.</p> <p style="text-align: right;">Full</p>
	<p>Deploy the components and files in the source package over the corresponding components and files in the existing package. When you choose to deploy package components, package files, or both, this is the default.</p> <p style="text-align: right;">Patch</p>
	<p>Note: Before you deploy a project, you can find out which assets Deployer will overwrite by generating the simulation report.</p>
Version	<p>Supply the version number to use for the source package in comparisons with existing packages on target Integration Servers.</p> <p>Whether Deployer actually deploys the package depends on the version numbers of the source package and the existing package. If the source package's version number is the same or higher than the existing package's version number, Deployer deploys. If the source package's version number is lower than the existing package's version number, Deployer does not deploy.</p> <p>Note: The version number for the source package on the source Integration Server is not affected by your entry here.</p>
Build	<p>Supply the build number to assign to the deployed package on the target Integration Servers.</p> <p>Note: To retain the patch history of a package on the target server, you must specify a build number for the package.</p>
Patches Included	<p>Supply the list of patches that have been applied to the deployed package on the target Integration Servers. Specify the patch numbers, separated by commas (for example, 44, 45, 55). Specify patches only if you selected Full for Package Type.</p>
Brief Description	<p>Supply a description to use for the deployed package on the target Integration Servers (for example, "December 2003 release with patches to correct Order Process</p>

Property	Entry
	problem.") Specify a description only if you selected Full for Package Type .

- In the **Recommendations for Target** area, you can recommend the minimum version of Integration Server and Java Virtual Machine (JVM) to run the source package. If the JVM version on the target Integration Server is lower than you specify here, Deployer will deploy the source package but will not activate it, regardless of the setting of the **Activate After Deployment** option. When this happens, the target Integration Server will display a warning about the JVM version. The defaults shown in this area reflect the Integration Server and JVM that host the source package.
- In the **Package Build Options** area, indicate whether Deployer should use the package version and build numbers that exist in the source Integration Server each time the user creates a build instead of the package version and build numbers specified in the *package_name* **Properties** area.
- In the **Package Deployment Options** area, specify the following:

Option	Entry
Activate After Deployment	How Deployer should deploy the package. Click: <ul style="list-style-type: none"> ■ Activate to enable the package. ■ Install Only to install the package but not enable it. ■ Inbound Only to neither install nor enable the package.
Sync Document Types	Whether Deployer should synchronize the publishable IS document types in the source package with documents types on the webMethods Brokers that are connected to the target Integration Servers.

Note: The connected webMethods Brokers must be available at deployment time for synchronization to occur. If a connected webMethods Broker is not available, IS document types are not synchronized for the Integration Server to which the webMethods Broker is connected. Deployer writes a message to that effect to the deployment report. Deployer can detect webMethods Broker unavailability when you generate the

Option	Entry
	simulation report and will write a message advising you of the problem to the report.
To...	Select...
Synchronize all publishable IS document types in the package that are new to the target Integration Servers. Do not synchronize IS document types in the package that already exist on the target Integration Servers, even if they have been modified.	New
Synchronize all publishable IS document types in the package.	All
Do not synchronize any IS document types.	None

- If you indicated in the project properties that you want Deployer to suspend individual triggers during deployment, click **Suspend Triggers**, select the check box next to each trigger to suspend, click **Suspend**, and then return to the previous page.
- If you indicated in the project properties that you want Deployer to suspend individual adapter notifications during deployment, click **Suspend Notifications**, select the check box next to each notification to suspend, click **Suspend**, and then return to the previous page.

Note: If you suspend a particular adapter notification but the notification does not exist on a target Integration Server, you will not be able to deploy. You can only suspend notifications that already exist on all target Integration Servers.

- Click **Save**.
- Repeat these steps for each package in the deployment set.
- If you are not going to add any more assets to the deployment set, go to ["Resolving Dependencies" on page 113](#).

Adding webMethods Files

When Deployer deploys a webMethods file, the file retains the read/write permissions it had on the source server.

For large projects, you might be able to stream webMethods files from the source server to Deployer and from Deployer to the target server. For detailed information on this option, see ["Setting General Deployment Defaults" on page 79](#).

To add Integration Server files

1. In the **Deployment Sets** area, under the deployment set to which to add webMethods files, click the **webMethods Files** folder. In the right-hand pane, Deployer lists the source Integration Servers you identified.
2. In the right-hand pane, open the tree to show the webMethods installation directory and its contents on the source Integration Servers. Select the check box next to each file to add to the deployment set.
3. Click **Save**. Deployer shows the selected assets in the left-hand pane under the **webMethods Files** folder for the deployment set.
4. If you are not going to add any more assets to the deployment set, go to ["Resolving Dependencies" on page 113](#).

Adding Trading Networks Assets

Note: Your source and target Integration Servers might have assets in common, and you therefore do not need to deploy them from one environment to another. You can improve Deployer performance by excluding these assets from deployment sets. For instructions, see ["Excluding Common Assets" on page 113](#).

To add Trading Networks assets

1. In the **Deployment Sets** area, under the deployment set to which to add Trading Networks assets, click the **Trading Networks** folder. In the right-hand pane, Deployer lists the source Integration Servers you identified.
2. In the right-hand pane, open the tree to show the Trading Networks assets on the source Integration Servers, then select the check box next to each asset to add to the deployment set.

Note: If you add a TN document type that is set up in Trading Networks for duplicate checking using custom services, Deployer does not detect the dependency on the service. If the service does not already exist on the target Integration Servers, you must add the service to the deployment set. If you do not, Deployer will log an error to the deployment report and will not deploy the TN document type.

3. Click **Save**. Deployer shows the selected assets in the left-hand pane under the **Trading Networks** folder for the deployment set.
4. If you are not going to add any more assets to the deployment set, go to ["Resolving Dependencies" on page 113](#).

Excluding Common Assets

Your source and target Integration Servers might have assets in common, and therefore you do not need to deploy them from one environment to another. For example, you might have created certain packages that exist on all your source and target Integration Servers.

Deployer lets you identify these common assets so they will not appear in the asset list when you define deployment sets or as referenced assets when you resolve dependencies (see "[Resolving Dependencies](#)" on page 113, below). The assets you specify on this list will be excluded for all IS & TN deployment sets in all projects. Excluding these common assets improves Deployer performance by reducing the amount of processing needed to produce the asset and dependencies lists, and by preventing you from deploying unnecessary assets.

You identify the common assets in a file named `common.cnf` in the `Integration Server_directory\WmDeployer\config` directory. By default, the file is pre-populated with the names of Integration Server packages you should never deploy to other Integration Servers using Deployer, but rather should only install on other Integration Servers using the Software AG Installer. The file also includes instructions, lists the asset types you can exclude, and shows examples. List the assets you want to exclude next to the appropriate asset types.

Resolving Dependencies

Deployer can determine when assets that are in a deployment set require other assets that are not in the deployment set. The assets that require other assets are called *dependent* assets, while the assets that are required are called *referenced* assets. Deployer identifies missing referenced assets as *unresolved dependencies*.

Deployment Set	Example of Unresolved Dependencies
webMethods Broker	If you add a client group but not the documents to which the client group can publish or subscribe, the documents are unresolved dependencies.
IS & TN	If you add a trigger but not the service that is invoked by the trigger, the service is an unresolved dependency.
MWS	If you add a page but not the portlets that are referenced by the page, the portlets are unresolved dependencies.

Deployment Set	Example of Unresolved Dependencies
Optimize	If you add a rule but not the dimensions used by the rule, the dimensions are unresolved dependencies.
ProcessModel	If you add a process model but not the flow services called by the process model, the flow services are unresolved dependencies.

In the project properties ("[Setting the Dependency Checking Default](#)" on page 79), you indicated how you want to check dependencies in the deployment sets. When Deployer automatically checks dependencies and finds unresolved dependencies in a deployment set, it shows  in the **Unresolved Dependencies** column for the deployment set; when there are no unresolved dependencies, Deployer shows  in the column. When you can check dependencies manually, Deployer shows  in the **Unresolved Dependencies** column for each deployment set; click **Check** next to the . If necessary, you can later "un-resolve" or remove a dependency you have resolved and resolve it again a different way.

To resolve dependencies

- In the **Unresolved Dependencies** column for the deployment set, click **Check**. Deployer shows all unresolved dependencies on the **Unresolved Dependencies** page. The **Referenced Assets** column lists the missing referenced assets. The next column offers the possible ways you can resolve the unresolved dependency. The **Asset** column shows the dependent assets.
- Tell Deployer how to resolve each unresolved dependency as described below. If you want to resolve all assets in a folder the same way, you can set the resolution at the folder level rather than at the level of the individual assets.

Option	Description
Add	If the referenced asset does not exist on the target servers and you want to deploy the referenced asset to them, use this option. Deployer adds the referenced asset to the deployment set. For Integration Server assets, you can choose to add the referenced asset or the entire package that contains the referenced asset.
Exists	If you believe the referenced asset already exists on the target servers and you want to continue working, but you want Deployer to make sure the asset does in fact exist later, use this option. Deployer will check for the referenced asset when you map the project to target servers. If Deployer does not find the asset, an icon alerts you during the mapping task. If you do not address the problem during the mapping task, Deployer will write a message about the problem to the

Option	Description
	simulation report. If you deploy without addressing the problem, Deployer will not deploy the deployment set.
Ignore	<p>If you want to bypass dependency checking for the referenced asset at this time so you can continue working, use this option. You might use this option if the referenced asset is missing on the source server. Missing referenced assets are marked with a question mark (?) on the Unresolved Dependencies page.</p> <p>Before deploying, make sure either that the referenced asset exists on the target server or that the referenced asset is unnecessary. If the referenced asset does not exist on the target server, Deployer might not be able to deploy correctly; if it can deploy, the deployed assets will not run correctly.</p> <p>Deployer will list ignored assets in the simulation report and in the deployment report.</p>
Unset	If you have set the assets in a folder to various settings and want to start over, use this option.

3. Click **Save**. Deployer moves dependencies you resolved using the **Exists** or **Ignore** option to the **Resolved Dependencies** page.
4. To see the resolved dependencies, click **Resolved Dependencies**.

You can un-resolve a resolved dependency and re-resolve it differently. To un-resolve a dependency, go to the **Resolved Dependencies** page, select the check box in the **Delete** column for the resolved dependency, and click **Delete**. Deployer returns the dependency to the **Unresolved Dependencies** page. Go to that page and re-resolve the dependency.

Manually Adding Dependencies to a Package Component in an IS & TN Deployment Set

Deployer cannot always detect all dependencies. If you are aware that an asset in an IS & TN deployment set has a dependency on a package component, and Deployer has not detected this dependency, you can manually *add* that dependency.

Deployer will check for the referenced asset when you map the project to target Integration Servers, as it does when you use the **Exists** option to resolve an unresolved dependency. If Deployer does not find the asset, an icon alerts you during the mapping task. If you do not resolve the dependency at that time, Deployer will write a message about it to the simulation report and, if you do not resolve it at that time, to the deployment report.

To manually add a dependency on a package component

1. Go to the **Resolved Dependencies** page as explained in the previous section.
2. Under the **Manually Add Dependency** area, in the **Referenced Package** box, type the name of the package that contains the referenced component.
3. In the **Referenced Component** box, type the name of the referenced component.
4. Click **Add**.

You can remove a dependency you added manually. To do so, return to the **Projects > project > Define** page, open the folder that contains the asset, navigate to the asset in the tree in the right-hand pane, cancel the selection of the asset by clearing the appropriate check box, and save the deployment set.

Removing Process Models from a Deployment Set

When you add a process model to a ProcessModel deployment set and then add referenced assets that reside on Integration Servers, Deployer shows the referenced assets as children of the process model. If you want to remove a process model from a deployment set, clear the check box next to the process model under the tree. This removes the process model from the deployment set; however, the dependencies must be removed manually.

Next Steps

Once you have selected assets for runtime-based deployment, you can do either of the following:

- Create a deletion set. See ["Defining a Deletion Set" on page 127](#).
- Build your project. See ["Building a Runtime-Based Deployment Project" on page 137](#).

8 Adding Assets for Repository-Based Deployment

■ Overview	118
■ Selecting Composites	118
■ Selecting Individual Assets from Composites	119
■ Resolving Dependencies	120
■ Resolving Conflicts	123
■ Deploying ACLs	124
■ Next Steps	124

Overview

In repository-based deployment, you can add only those assets from composites that are present in the repository. The repository can contain several composites, and you can deploy assets from any composite in the repository, but you cannot add assets from more than one repository to one deployment set. Assets in one composite in the repository can have dependencies on one or more assets in other composites in the repository.

If you are adding assets for Integration Server, Trading Networks, or Broker, you have the option of adding individual assets from a composite to your deployment set, instead adding the entire composite.

Before you can add assets for repository-based deployment, you must perform the following:

1. Build the composites and add a source repository. For more information, see ["Building Composites for Repository-Based Deployment" on page 29](#).
2. Configure a source repository. For more information, see ["Connecting to a Repository for Repository-Based Deployment" on page 72](#).
3. Connect to target servers, target groups, or both. For more information, see ["Starting Deployer and Connecting to Servers" on page 53](#).
4. Define a deployment set. For more information, see ["Defining a Deployment Set" on page 95](#).

Selecting Composites

Perform the following to select entire composites from the repository.

To select composites for a deployment set

1. In the **Deployment Sets** area of the **Deployer > Projects > project > Define** page, click  for every repository for which you want to add assets.

In the right-hand pane, Deployer displays the  icon for runtime types contained in the repository you selected.

2. Click the plus sign to expand the  runtime types to display the composites they contain.

Deployer displays the  icon for composites.

3. Click every composite whose assets should be part of the deployment set.

To select all of the composites for the runtime type, click . To select all of the composites available on the repository, click .

4. Click **Save**.

Deployer adds the runtime type and the child composites as children of the deployment set in the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.

Selecting Individual Assets from Composites

For Integration Server, Trading Networks, and Broker (including JNDI) assets, you can choose to add individual assets from composites rather than the entire composite. Adding only some of the assets to the deployment set instead of the entire composite is referred to as *partial deployment*.

Note: For the partial deployment of JNDI assets, export JMS destinations into one JNDI context at a time. Partial deployment of JMS destinations into multiple JNDI contexts is not supported.

Perform the following to select individual assets from composites.

To select individual assets for a deployment set

1. In the **Deployment Sets** area of the **Deployer > Projects > project > Define** page, click  for every repository for which you want to add assets.

In the right-hand pane, Deployer displays the  icon for runtime types contained in the repository you selected.

2. Click the plus sign to expand the  runtime types to display the composites they contain.

Deployer displays the  icon for composites.

3. Click the name of the composite from which you want to select individual assets.

Deployer displays the composite in the **Select Components** area of the right hand pane.

4. Click the plus sign to expand the  composite to reveal the assets it contains.
5. Click the assets that should be included in the deployment set.
6. Click **Save**.

Deployer displays the  icon for a partial composite with the assets you selected as its children in the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.

Resolving Dependencies

Deployer can determine when assets that are in a composite require other assets. The assets that require assets from other composites are called *dependent* assets, while the assets that are required are called *referenced* assets.

Deployer identifies missing referenced assets as follows:

- *Unresolved dependencies* are those dependencies that are present in the repository, but not in the deployment set. Deployer enables you to add, ignore, or automatically resolve those dependencies. Deployer checks dependencies automatically and shows one of the following in the **Unresolved Dependencies** column for the deployment set:

Deployer shows...



When an asset in a composite contains...

Unresolved dependencies. You can resolve dependencies automatically or manually.

- For information about resolving dependencies automatically, see ["Resolving Dependencies Automatically" on page 120](#).
- For information about resolving dependencies manually, see ["Resolving Dependencies Manually" on page 121](#).



No unresolved dependencies.

- *Missing dependencies* are those dependencies that are not available in the repository, so you cannot add them to your deployment set. You can set Deployer to ignore missing dependencies when you create the project (see ["Creating a Project" on page 85](#)) or when you check unresolved dependencies. For more information about setting missing dependency options while checking for unresolved dependencies, see ["Resolving Dependencies Manually" on page 121](#).

Resolving Dependencies Automatically

Perform the following procedure to set Deployer to resolve dependencies automatically.

To set Deployer to resolve dependencies automatically

1. In the **Unresolved Dependencies** column for the deployment set, click **Check**.
Deployer displays the **Unresolved Dependencies** page.
2. Click one of the following:

Option	Description
Auto resolve by Composite	<p>Deployer automatically resolves all the unresolved dependencies for the composite in the deployment set. Deployer checks for the dependencies in the repository. If the dependent composites are available in the repository, Deployer adds the composites to the deployment set.</p> <p>If the referenced composites are not available in the repository, Deployer cannot add the composites to the deployment set. You can then choose to ignore the missing dependencies.</p>
Auto resolve by Asset	<p>Deployer automatically resolves the <i>partial</i> addition of composite at the asset level. For example, if AssetA is dependent on AssetB in CompositeB, then Deployer adds only AssetB, instead of the entire composite (CompositeB).</p> <p>If the referenced assets are not available in the repository, Deployer cannot add the composites to the deployment set. You can then choose to ignore the missing dependencies.</p>

Deployer lists the missing dependencies on the **Missing Dependencies** page when you click **Check** in the **Unresolved Dependencies** column. You cannot add missing dependencies to the deployment set.

3. To set Deployer to ignore the missing dependencies perform the following:
 - a. Click **Ignore Missing Dependencies (Project Level)**.
 - b. Click **Apply**.
4. Click **Save**.

Resolving Dependencies Manually

Perform the following procedure to resolve dependencies manually.

To manually resolve dependencies

1. In the **Unresolved Dependencies** column for the deployment set, click **Check**.

Deployer shows all unresolved dependencies on the **Unresolved Dependencies** page as follows:

- The **Referenced Asset Composites** column lists the missing referenced assets.

- The **Unset/Add/Ignore** column offers the possible ways you can resolve the unresolved dependency.
 - The **Assets** column shows the dependent assets.
2. Click one of the following options to set how Deployer should resolve each unresolved dependency. If you want to resolve all assets in a composite the same way, you can set the resolution at the composite level rather than at the level of the individual assets.

Option	Description
Unset	This is the default status. If you click Unset after you have made another selection (Add or Ignore) Deployer resets the assets to an unresolved status. Use this option if you set the assets in a composite to either Add or Ignore and want to start over.
Add	<p>Adds the referenced asset to the deployment set. Use this option if the referenced asset does not exist on the target server and you want to deploy the referenced asset to it.</p> <p>For Integration Server, Trading Networks, or Broker (including JNDI) assets, you can choose to add the referenced asset or the entire composite that contains the referenced asset. Adding only the referenced asset to the deployment set instead of the entire composite is referred to as <i>partial deployment</i>.</p> <p>Note: For the partial deployment of JNDI assets, export JMS destinations into one JNDI context at a time for partial deployment. Partial deployment of JMS destinations into multiple JNDI contexts is not supported.</p>
Ignore	<p>Ignores the asset. Use this option if you want to bypass dependency checking for the referenced asset so you can continue working.</p> <p>Note: Before deploying, make sure that either the referenced asset exists on the target server or that the referenced asset is unnecessary. If the referenced asset does not exist on the target server, Deployer might not deploy correctly; if it can deploy, the deployed assets will not run correctly.</p>

3. Click **Save**.
- Deployer moves dependencies you resolved using the **Add** option to the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand

pane. If you added only a subset of Integration Server, Trading Networks, or Broker (including JNDI assets) assets rather than an entire composite (partial deployment), Deployer displays  as a sibling of .

- Deployer adds dependencies you resolved using the **Ignore** option to the **Name** column of the **Deployer > Projects > project > Define** page, but the dependency remains listed on the **Unresolved Dependencies** page.

Resolving Conflicts

A *conflict* occurs when a dependent asset is available in multiple composites or when different assets in one composite share the same asset name. Deployer displays conflicts on the same page as unresolved assets.

To resolve conflicts

1. In the **Unresolved Dependencies** column for the deployment set, click **Check**.

Deployer shows all conflicted assets on the **Unresolved Dependencies** page as follows:

- The **Referenced Conflict Asset Composites** column lists the assets in conflict.
- The **Unset/Add/Ignore** column offers the possible ways you can resolve the conflict.
- The **Assets** column shows the dependent assets.

2. Click one of the following options to set how Deployer should resolve each conflict.

Option	Description
Unset	This is the default status. If you click Unset after you have made another selection (Add or Ignore) Deployer resets the assets to an conflicted status. Use this option if you set the assets in a composite to either Add or Ignore and want to start over.
Note: After you use Unset to clear your previous selection, you must set the asset to either Add or Ignore .	
Add	Adds the referenced assets to the deployment set. Use this option to deploy the referenced assets.
Ignore	Ignores the conflict. Use this option to ignore the referenced assets.

3. Click **Save**.

- Deployer moves dependencies for which conflicts were resolved using the **Add** option to the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.
- Deployer ignores the conflicts for those assets which you set to **Ignore**.

Deploying ACLs

If you want to deploy ACLs, you must perform extra steps depending on whether the ACLs are associated with My webMethods Server groups or LDAP groups.

Deploying ACLs Associated with My webMethods Server Groups

If you want to deploy ACLs that are associated with My webMethods Server groups, you must perform the following steps.

To deploy ACLs associated with My webMethods Server groups

1. Use an MWS deployment set to deploy the My webMethods Server groups to the targetIntegration Servers.
2. Create an IS & TN deployment set containing the ACLs.
3. Mark the unresolved dependencies for the My webMethods Server groups as **Ignore**.
4. Deploy the ACLs.

Deploying ACLs Associated with LDAP Groups

If you want to deploy ACLs that are associated with LDAP groups, perform the following steps.

To deploy ACLs associated with LDAP groups

1. Configure the LDAP groups on the targetIntegration Servers.
2. Create an IS & TN deployment set containing the ACLs.
3. Mark the unresolved dependencies for the LDAP groups as **Ignore**.
4. Deploy the ACLs.

Next Steps

Once you have selected assets for repository-based deployment, you can perform the following.

To...	See...
Add deletion sets to the project	"Defining a Deletion Set" on page 127
Map the project	"Mapping a Project" on page 141
Deploy the project	"Deploying a Project" on page 151

9 Defining a Deletion Set

■ About Deletion Sets	128
■ Creating a Deletion Set	128
■ Identifying Servers	130
■ Adding Assets to a Deletion Set	131
■ Resolving Dependencies in Repository-Based Deletion Sets	134
■ Exporting and Importing Deletion Set Definitions	134
■ Next Steps	135

About Deletion Sets

You use *deletion sets* to identify those assets you want to delete from the target server during deployment. When you deploy the project, Deployer deletes the assets defined in the deletion set from the target server and then deploys assets defined for the deployment set from the source server to the target server.

Keep the following points in mind when working with deletion sets:

- Deployer supports deletion sets for both runtime and repository-based projects.
- If Deployer does not display a server you want to use as a target, you have not yet set it up to work with Deployer. For more information about defining target servers, see ["Starting Deployer and Connecting to Servers"](#) on page 53.
- For runtime-based deployment, Deployer supports deletion sets for Broker, Integration Server, and Trading Networks assets only. For repository-based deployment, Deployer supports deletion sets for all runtimes.

Defining deletion sets involves the following stages:

Stage	Procedure
Stage 1	Create a deletion set. See "Creating a Deletion Set" on page 128.
Stage 2	Identify target servers from which to delete the assets. See "Identifying Servers" on page 130.
Stage 3	Add the assets to the deletion set. See "Adding Assets to a Deletion Set" on page 131.
Stage 4	For repository-based deployment sets only, resolve dependencies. See "Resolving Dependencies in Repository-Based Deletion Sets" on page 134.
Stage 5	For runtime-based projects, you can export deletion set definitions from one project and import them into another. See "Exporting and Importing Deletion Set Definitions" on page 134.

Creating a Deletion Set

To create a deletion set, perform the following.

To create a deletion set

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Define**.
5. Click **Create Set**.

Deployer displays the Create Set properties in the right-hand pane.

6. Complete the following fields:

Box	Entry
Type	(Runtime-based deployment only.) Runtime type of the server that contains the assets to delete.
Set	Deletion
Name	Name to use for the deletion set. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Description	Description for the deletion set. The description length has no limit and can include any characters.
Maximum TN Assets to Display	(Runtime-based deployment only.) Number of Trading Networks assets Deployer should display. Depending on your browser, Deployer might not be able to display more than 1000 assets for Trading Networks. If the source server hosts more than 1000 assets, use this field with the with the All other assets field to reduce the number of assets displayed.
Packages (IS & TN deletion set)	(Runtime-based deployment only.) After you choose the servers from which to define deletion sets, Deployer displays all packages on the servers. You can use this field to narrow the display. Type a regular expression that specifies the text that the package names must contain in order to be listed.
All other assets	(Runtime-based deployment only.) After you choose the servers from which to define deletion sets, Deployer will display all assets on those servers. You can use this field to

Box	Entry
	<p>narrow the display. Specify a regular expression that specifies the text that the asset names must contain in order to be listed.</p> <p>Note: Deployer can display up to 10,000 assets. If the source server hosts more than 10,000 assets, use the Packages and All other Assets fields to reduce the number of assets to be displayed.</p>

7. Click **Create**.

Deployer displays the deletion set in the left-hand pane in the **Deletion Sets** area.

8. Perform the tasks in "[Identifying Servers](#)" on page 130 to identify the servers that contain the assets to add to the deletion set.

Identifying Servers

To select the assets to add to the deletion set, you must first identify the servers that contain those assets.

Deployer lists the version of each target server in the **Version** column. You must select target servers with the same version for the deletion set. You cannot include target servers with different versions in a deletion set. For information about selecting the versions of target servers, see "[Connecting to webMethods Servers](#)" on page 54.

To identify servers for a deletion set in a runtime-based project, see "[Identifying Servers for a Runtime-Based Deletion Set](#)" on page 130. To identify servers for a deletion set in a repository-based project, see "[Identifying Servers for a Repository-Based Deletion Set](#)" on page 131.

Identifying Servers for a Runtime-Based Deletion Set

Perform the following procedure to identify the servers for a deletion set in a runtime-based project.

To identify servers for a runtime-based project

1. On the **Deployer > Projects > *project* > Define** page, in the left-hand pane in the **Name** column, click the deletion set for which to identify servers.
2. In the **Select Servers** column in the right-hand pane, select the check box next to each server that contains assets to add to the deletion set.
3. Click **Save**.
4. Perform the tasks in "[Adding Assets to a Deletion Set](#)" on page 131 to add assets to the deletion set.

Identifying Servers for a Repository-Based Deletion Set

Perform the following procedure to identify the servers for a repository-based project.

To identify servers for a repository-based project

1. On the **Deployer > Projects > project > Define** page, in the left-hand pane in the **Name** column of the **Deletion Sets Repository** area, click the deletion set for which to identify servers.
2. In the **deletion_set > Select Server** area in the right-hand pane, select the runtime type of the server to include in the deletion set from the **Select Server Type** list.

Deployer displays a list of all servers of the specified type that are set up to work with your system.

3. Select the check box next to each target server that contains assets to add to the deletion set.
4. Click **Add**.

Deployer displays the server you added to **Deletion Sets** area in the left-hand pane.

5. Perform the tasks in "[Adding Assets to a Deletion Set](#)" on page 131 to add assets to the deletion set.

Adding Assets to a Deletion Set

You can choose assets to add to a deletion set from any server that is similar to the target server from which you want to actually delete the assets. Keep the following points in mind when adding assets to a deletion set:

- For Trading Networks, you cannot delete document attributes, field definitions, binary types, or profile security data.
- When you map a deletion set to target servers, Deployer identifies assets that depend on the assets you want to delete and lets you resolve those dependencies (see "[Mapping a Project to Target Servers and Target Groups](#)" on page 142). However, Deployer can only detect dependencies among assets from the same type of product (for example, among Integration Servers). It cannot detect dependencies among assets from different products (for example, among Integration Servers and ProcessModel servers). Make sure that assets you want to delete for one type of product are not required by assets of other types of product.

Adding Assets to a Runtime-Based Deletion Set

Perform the following procedure to add assets to a runtime-based deletion set.

To add assets to a runtime-based deletion set

1. In the **Deletion Sets** area, under the deletion set to which to add assets, click the folder that contains assets you want to add to the deletion set.

In the right-hand pane, Deployer lists the servers you identified in "[Identify Source Servers for the Deployment Set](#)" on page 97.

Note: For information about adding full or partial packages to a deletion set, see "[Adding Packages to Deletion Sets](#)" on page 132.

2. In the **Select** column of the **Select Servers** area, select the check box next to each source server that contains assets to add to the deployment set.

Deployer lists the version of each source server in the **Version** column. You must select servers with the same version for the deletion set. You cannot include servers with different versions in a deletion set. For information about selecting the versions of source servers, see "[Connecting to webMethods Servers](#)" on page 54.
3. In the right-hand pane, open the tree to show the assets on the servers, select the check box next to each asset to add to the deletion set, and then click **Save**.

Deployer shows the selected assets in the left-hand pane under the folder you clicked in the previous step.

Adding Assets to a Repository-Based Deletion Set

Perform the following procedure to add assets to a repository-based deletion set.

To add assets to a repository-based deletion set

1. In the **Deletion Sets** area, under the deletion set to which to add assets, click the server () that contains assets you want to add to the deletion set.
2. In the **Select Assets or Asset Components** area in the right-hand pane, expand the tree to display the assets on the servers and select the check box next to each asset to add to the deletion set.
3. Click **Save**.

Deployer displays the assets you selected in the left-hand pane in the **Deletion Sets** area. The assets are grouped by asset categories. For example, Integration Server assets are grouped by isfile, ispackages, and so on.

Adding Packages to Deletion Sets

If you are creating an IS & TN deletion set, you can add Integration Server packages in their entirety, or you can add selected package components only (called *partial packages*). You can add full or partial packages to either runtime or repository-based projects.

To add full or partial packages to deletion sets

1. Perform one of the following:

<u>If your project is...</u>	<u>Do this...</u>
Runtime-based	In the Deletion Sets Runtime area, under the deletion set to which to add packages or partial packages, click the Packages folder.
Repository-based	In the Deletion Sets area, under the deletion set to which to add the packages, click the server  that contains packages or partial packages.

In the right-hand pane, Deployer lists the Integration Servers you identified in ["Identify Source Servers for the Deployment Set" on page 97](#).

2. In the right-hand pane, expand the tree to display the packages on the Integration Servers, then do one of the following:

<u>To add...</u>	<u>Do this...</u>
Full packages	Select the check boxes next to the packages to add in their entirety to the deletion set and click Save . Deployer shows the entire package icon  for the selected packages in the left-hand pane under the Packages folder (for runtime-based deployment) or under the server  (for repository-based deployment).
Partial packages	<ol style="list-style-type: none"> a. Click the package name. b. In the Select Components area, open the tree to show the package components, then select the check box next to each component to add to the deletion set. c. Click Save. d. Click Return to Packages. Deployer shows the partial package icon  for the package in the left-hand pane under the Packages folder (for runtime-based deployment) or under the server  (for repository-based deployment).

Note: If you add a partial package and later want to include the entire package instead, cancel the selection of the components by clicking the name of the partial package, clearing all checked boxes, and clicking **Save**. Then save the deletion set and add the entire package as explained above.

Resolving Dependencies in Repository-Based Deletion Sets

For repository-based projects, Deployer checks dependencies in deletion sets automatically and shows one of the following in the **Unresolved Dependencies** column for the deployment set:

Deployer shows...	When a deletion set contains...
	Unresolved dependencies. Note: You must resolve unresolved dependencies or deployment will fail.
	No unresolved dependencies. Note: Deployer does not check dependencies in deletion sets for runtime-based projects. For a full explanation of unresolved dependencies for a repository-based project, see "Resolving Dependencies" on page 120 .

Perform the following procedure to set Deployer to resolve dependencies for repository-based deletion sets.

To resolve dependencies for a repository-based deletion set

1. In the **Unresolved Dependencies** column for the deployment set, click  **Check**.
Deployer displays the **Unresolved Asset References** page.
2. Perform one of the following:
 - a. To automatically resolve all unresolved dependencies for the composite, click **Auto resolve all missing references**.
 - b. To manually resolve unresolved dependencies, select the check box in the **Add** column for each referenced asset to add to the deletion set and click **Add**.

Exporting and Importing Deletion Set Definitions

For runtime-based projects, Deployer exports and imports deletion set definitions separately from the rest of the project properties exported and imported in ["Exporting and Importing Project Properties" on page 89](#).

Note: You cannot export and import deletion set definitions for repository-based projects.

When you export deletion set definitions, Deployer creates a file called *project_deleteSets.xml* that contains the deletion set definitions. This file is stored in the

Integration Server_directory\instances*instance_name* \packages\WmDeployer\replicate\outbound directory. Deployer also gives you the option to save the file to your local file system. You can then use this file to import the deletion set definitions into another Deployer project.

To export and import deletion set definitions

1. Export deletion set definitions from a project as follows:
 - a. In Deployer, go to the **Deployer > Projects** page.
 - b. In the **Name** column, click the project from which to export.
 - c. In the right-hand pane, click  **Define**.
 - d. Click **Export Deletion Set Definitions**.
2. Import deletion set definitions into a project as follows:
 - a. Copy the *project_deleteSets.xml* file to the *Integration Server_directory*\instances*instance_name* \packages\ WmDeployer\replicate\inbound directory on the machine that hosts the project.
 - b. In Deployer, go to the **Deployer > Projects** page.
 - c. In the **Name** column, click the project into which to import.
 - d. In the right-hand pane, click  **Define**.
 - e. Click **Import Deletion Set Definitions**, then select the *project_deleteSets.xml* file you just copied to the inbound directory.

Next Steps

Once you have created a deletion set you can perform the following:

For...	See...
Runtime-based projects, you can build your project.	"Building a Runtime-Based Deployment Project" on page 137
Repository-based projects, you can map your project.	"Mapping a Project" on page 141

10 Building a Runtime-Based Deployment Project

■ Creating a Build	138
■ Rebuilding a Build	139
■ Exporting and Importing a Build	139
■ Next Steps	140

Creating a Build

To create a build

1. In **Deployer**, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. Click  **Build**. Deployer displays the **Projects > project > Build** page and lists all builds that exist for the selected project.

The **Status** column on the **Projects > project > Build** page indicates whether each project build is in sync with the current project definition. If the build and the current project definition are in sync, the column shows . If the project definition has changed since the build was created, the column shows . You can rebuild such a project if you want. For instructions, see ["Rebuilding a Build" on page 139](#).

To see the progress report of the current or last action, click  in the **Progress Report** column. The progress report displays the updates for build requests as they occur. This is useful in the case where the deployment build takes a long time to finish.

5. In the left-hand pane, click **Create Build**.
6. In the **Name** box accept the default build name or replace it with a name that you choose. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:
`$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "`
7. In the **Description** box, you can type a description for the build. The description can be of any length and can include any characters.
8. Click **Create**.

Note: If the project for which you are trying to create the build contains unresolved dependencies, you will receive a message to that effect and the build process will fail. For instructions on displaying and resolving unresolved dependencies, see ["Resolving Dependencies" on page 113](#).

9. To view the progress of the build, click the **View Progress Report** link. The progress report displays the updates for build requests as they occur. This is useful in the case where the deployment build takes a long time to finish.
10. Under **Build History** in the right-hand pane, click  in the **Report** column to display the build report in HTML or XML.

The build report lists the assets that were successfully included in the build, describes any errors that occurred during the build process, and informs you if the project contains unresolved dependencies. The report is also available under

the name `BuildReport_reportID.xml` in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\pub\projects\project_name\builds\build_name\reports` folder, where `project_name` is the name of the project and `build_name` is the name of the build.

Rebuilding a Build

The **Status** column on the **Projects > project > Build** page indicates whether each project build is in sync with the current project definition. If the build and the current project definition are in sync, the column shows . If the project definition has changed since the build was created, the column shows .

If a project build is out of sync with the current project definition or contains assets that you know have changed on the source servers, and you want to re-create the build to bring it up to date, click  in the **Rebuild** column for the build.

If you want to see the progress report, click  in the **Progress Report** column. The progress report displays the updates for build and rebuild requests as they occur. This is useful in the case where the deployment build is large and it takes a long time to finish.

Note: Deployer does not take dependencies into account while rebuilding project builds.

If the project for which you are trying to create the build contains unresolved dependencies, you will receive a message to that effect and the build process will fail. When asset dependencies are changed on the source server, you must first remove the dependent asset from the deployment set and save the deployment set. Then, add the asset back and save the deployment set. When Deployer displays the changed dependencies, resolve the dependencies and rebuild the project. For instructions on displaying and resolving unresolved dependencies, see ["Resolving Dependencies" on page 113](#).

Exporting and Importing a Build

You can export the build you want to deploy from the Deployer in one environment and import the build into the Deployer in another environment. The Deployer into which you import the build automatically creates the deployment project and deployment sets from the imported build. You can then map the imported build, or you can export a deployment map for the build from the Deployer in the source environment and import it into the target project. For more information about importing and exporting maps, see ["Exporting and Importing a Map" on page 145](#).

To export and import a build

1. Export a build as follows:
 - a. In the sourceDeployer, go to the **Deployer > Projects > project > Build** page.

- b. Locate the build to export and click  in the build's **Export** column. Deployer creates a file that contains the build. The file is named *projectName_ExportedBuild_buildName* and is stored in the *Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\outbound* directory. Deployer also allows you to save the file to your local file system.
 - c. If you have previously exported a build of the same name, Deployer displays a dialog box confirming that you want to overwrite the existing build. Click **OK** to overwrite the existing build.
2. Import the build as follows:
 - a. Copy the *projectName_ExportedBuild_buildName* file to the *Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound* directory on the machine that hosts the target Deployer.
 - b. In the target Deployer, go to the **Tools > Import Build** page.
 - c. In the **Project Build** list, click the *projectName_ExportedBuild_buildName* file you just copied to the inbound directory.
 - d. Click **Import**.

Next Steps

Once you have created a build, you can map and deploy the project.

<u>To...</u>	<u>See...</u>
Map the project	"Mapping a Project" on page 141
Deploy the project	"Deploying a Project" on page 151

11 Mapping a Project

■ About Mapping a Project	142
■ Mapping a Project to Target Servers and Target Groups	142
■ Exporting and Importing a Map	145
■ Substituting Configuration Values	146
■ Exporting and Importing Substitute Configuration Values	148

About Mapping a Project

Mapping a project involves the following tasks. Unless otherwise noted, you can perform all tasks for both runtime-based and repository-based deployment projects.

- Map a project to target servers and target groups. For more information, see ["Mapping a Project to Target Servers and Target Groups" on page 142](#).
- Export a deployment map from one project and import it into another. For more information, see ["Exporting and Importing a Map" on page 145](#).
- Specify configuration values to substitute for assets. For more information, see ["Substituting Configuration Values" on page 146](#).
- Export substitute configuration values from one deployment map and import them into another. For more information, see ["Exporting and Importing Substitute Configuration Values" on page 148](#).

Mapping a Project to Target Servers and Target Groups

You can map a project to individual target servers, target groups, or both.

To map a project to target servers

1. In **Deployer**, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Map**. **Deployer** displays the **Projects > project > Map** page and lists all maps that exist for the selected project.
5. In the left-hand pane, click **Create Deployment Map**.
6. In the **Name** box, accept the default deployment map name or replace it with a name that you choose. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:
`$ ~ / \ # & @ ^ ! % * : ; , + = < ' ' "`
7. In the **Description** box, type a description for the map. The description length has no limit and can include any characters.
8. Click **Create**.
9. Under the **Deployment Map Topology** area, in the **Set Mapping** column for a deployment set, map to the target servers to which to deploy the assets, as follows:

To add a...	Do this...
Individual target server	<p>Click Add Target Server and then perform one of the following:</p> <ul style="list-style-type: none"> ■ For runtime-based projects, select the check box next to each target server to which to deploy the assets in the deployment set and then click Add. ■ For repository-based projects: <ol style="list-style-type: none"> i. In the Select Server list, click the runtime type of the target server. ii. Select the check box next to each target group to which to deploy the assets in the deployment set and then click Add. <p>Notes:</p> <ul style="list-style-type: none"> ■ For runtime-based projects, Deployer lists only those servers running compatible versions for selection as target servers. Repository-based deployment does not support mapping to target servers of versions that are different than the source repository. For information about selecting the versions of source servers, see "Connecting to webMethods Servers" on page 54. ■ If you are mapping a repository-based project that contains BPM ProcessModels, you must also select the physical Integration Server servers or My webMethods Servers for each logical server in the deployment set from the Map Logical Servers area of the Add Targets pane. ■ If a server you want to map to does not appear in the list, you have not yet set it up to work with Deployer. For instructions, see "Starting Deployer and Connecting to Servers" on page 53. Then click Refresh this Page to update the list of servers on this page.
	<p>Note: When you deploy Trading Networks assets, Deployer updates the Trading Networks database with the deployed assets. If Trading Networks is installed on multiple Integration Servers, map deployment sets that contain Trading Networks assets to only one of the Integration Servers. Do not map to multiple Integration Servers or you will experience unpredictable results when you deploy.</p>

To add a...	Do this...
Target group	<p>Click Add Target Group and then perform one of the following:</p> <ul style="list-style-type: none"> ■ For runtime-based projects, select the check box next to each target group to which to deploy the assets in the deployment set and then click Add. ■ For repository-based projects: <ol style="list-style-type: none"> i. In the Select Server list, click the runtime type of the target server. ii. Select the check box next to each target group to which to deploy the assets in the deployment set and then click Add. <p>Notes:</p> <ul style="list-style-type: none"> ■ For runtime-based projects, Deployer lists only those target groups whose version is compatible with the source servers in the deployment set. Repository-based deployment does not support mapping to target servers of versions that are different than the source repository. For information about selecting the versions of source servers, see "Connecting to webMethods Servers" on page 54. ■ If you are mapping a repository-based project that contains BPM ProcessModels, you must also select the physical Integration Server servers or My webMethods Server target groups for each logical server in the deployment set from the Map Logical Servers area of the Add Targets Groups pane.

10. If you are mapping a runtime-based deployment deletion set, under the **Deployment Map Topology** area, in the **Set Mapping** column, follow the instructions in the previous step, but map to the target servers from which to delete the assets. The important note is also true for deletion sets.
11. When Deployer returns to the *map* > **Properties** page, the **Deployment Map Topology** area might show  or .
 - For repository-based deployment projects, Deployer verifies whether the target servers or target groups are available for deployment when you add them to the deployment map. The **Status** column shows  if the server is available for deployment and  if it is not.
 - For runtime-based deployment sets,  appears in the **Referenced Assets** column and indicates that you resolved an unresolved dependency using the **Exists** option, but Deployer has found that the referenced asset does *not* exist on target

servers. Click  to see the missing referenced asset. You can then place the referenced asset on the target servers, or you can return to the project definition stage and re-resolve the dependency in a different way. For more information, see ["Resolving Dependencies" on page 113](#) (for runtime-based deployment).

- For deletion sets, dependencies work in the opposite direction from deployment sets. Deployer finds all assets on the target servers that depend on assets in the deletion set. If you were to delete the assets in the deletion set from the target servers, the dependent assets would no longer work properly. On the **map >Properties** page, in the **Deployment Map Topology** area, therefore, the  icon appears in the **Dependent Assets** column, and indicates that dependent assets exist. Click  to see the dependent assets, then choose whether to **Add** the dependent assets to the deletion set or to **Remove** the assets they depend on from the deletion set.

Note: Keep the following points in mind when working with dependencies:

- Deployer cannot detect dependencies across products. Make sure assets you want to delete are not required by assets of other products.
- If you do not address problems at this time, Deployer will write messages about them to the simulation report. If you deploy without addressing problems, Deployer will not deploy the assets identified in the deployment set or delete the assets identified in the deletion set.

12. If you are mapping a runtime-based deployment or deletion set and you resolved dependencies in the previous step, the contents of the deletion set have changed. As a result, you must rebuild the project (see ["Rebuilding a Build" on page 139](#)). If you exported the deletion set definition or the project build, you must also re-export the definition (see ["Exporting and Importing Deletion Set Definitions" on page 134](#)) and the build (see ["Exporting and Importing a Build" on page 139](#)).

Exporting and Importing a Map

You can export the deployment map from one Deployer environment and import to another Deployer environment. For example, you can export a map from the Deployer in your development environment to the Deployer in your testing environment. You must ensure that all target aliases in the test Deployer are the same as those in the development Deployer.

Before you import a map, you can edit any of the attributes (for example, you could map a deployment set to a different target server).

To export and import a map

1. Export a map as follows:
 - a. In the sourceDeployer, go to the **Deployer > Projects >project > Map** page.

- b. Locate the map to export and click  in the map's **Export** column. Deployer creates a file that contains the deployment map. The file is named `project_map.map` and is stored in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\outbound` directory. Deployer also allows you to save the file to your local file system.
2. After you export a map, you can edit any of the attributes before importing it into the target environment. For example, you might want to map a deployment set to a new target server or target group. For instructions, see ["Editing a Deployment Map, Project Properties, or Substitute Configuration Values" on page 179](#).
3. Import the map as follows:
 - a. Copy the `project_map.map` file to the `Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound` directory on the machine that hosts the target Deployer.
 - b. In the target Deployer, go to the **Deployer > Projects > project > Map** page.
 - c. Click **Import Map**, then select the `project_map.map` file you just copied to the inbound directory.

Substituting Configuration Values

Some assets might be configured differently on the source server (for runtime-based deployment) or repository (for repository-based deployment) than on the target server. You use Deployer to substitute different configuration values for assets during deployment so the assets will run properly on the target servers.

For example, as part of the deployment map for an IS & TN deployment set, you can specify configuration values for Integration Server assets that you want Deployer to substitute during deployment so the assets will run properly on target servers. Suppose an Integration Server in a development environment has a file polling port that is configured to monitor the `C:\TEMP` directory. You want to deploy this port to a production Integration Server on a Solaris system and have the port poll the `/tmp` directory instead. In the deployment map, you would specify a substitute configuration value of `/tmp` directory for the port. You can substitute different configuration values for scheduled tasks, ports, adapter connections, adapter notifications, and extended settings. You can substitute different configuration values for different target servers.

You can substitute configuration values as follows:

If you are creating a...

Runtime-based deployment project

You can substitute configuration values by...

Asset or target server. See ["Substituting Configuration Values by Asset" on page 147](#) and ["Substituting Configuration Values by Target Server \(Runtime-Based\)" on page 147](#).

If you are creating a...

Repository-based deployment project

You can substitute configuration values by...

Target server. See ["Substituting Configuration Values by Target Server \(Repository-Based\)"](#) on page 148.

Substituting Configuration Values by Asset

Note: You can substitute configuration values by asset only in runtime-based deployment.

Perform the following steps to substitute configuration values by asset for runtime-based deployment projects.

To substitute configuration values by asset

1. Under the **Deployment Map Properties** area, click **Configure Builds by Assets**. Deployer lists assets that have configuration values in the left-hand pane.
2. Substitute different configuration values for an asset as follows:
 - a. In the left-hand pane, click the asset. Deployer displays the asset's configuration values as they exist on the source server.
 - b. In the right-hand pane, type the configuration values to substitute.
 - c. In the bottom right-hand pane, select the target servers or target groups on which to make the substitutions.
 - d. Click **Save Substitutions**.

Substituting Configuration Values by Target Server (Runtime-Based)

Perform the following steps to substitute configuration values by target server for runtime-based deployment projects.

To substitute configuration values by target server for a runtime-based project

1. Under the **Deployment Map Properties** area click **Configure Builds by Server**. Deployer lists the target servers that are mapped to the deployment set.
2. Select a target server. Deployer lists assets that have configuration values in the right-hand pane.
3. Substitute different configuration values for an asset as follows:
 - a. In the right-hand pane, click the asset. Deployer displays the asset's configuration values as they exist on the source server.
 - b. In the bottom right-hand pane, type the configuration values to substitute.

- c. Click **Save Substitutions**.

Substituting Configuration Values by Target Server (Repository-Based)

Perform the following steps to substitute configuration values by target server for repository-based deployment projects.

To substitute configuration values by target server for a repository-based project

1. On the **project > Map** page, in the **Configured** column, click .

Deployer opens a new page that displays target servers and target groups that are mapped to the deployment set.
2. In the **project > deployment map > Target Servers** pane, select the target servers for which you want to substitute values.

Deployer lists the composites that have configuration values in the center **Configurable Composites** pane.

3. Click the composite for which you want to substitute asset values.

Deployer displays the assets in the right-hand **Configurable Components** pane.
4. Click the assets in the **Name (Implementation Type)** column of the **Configurable Components** pane.

Note: You can select more than one asset at a time.

Deployer displays the configuration values as they exist on the repository in the **Target Substitutions and Source Values** pane. If you selected multiple assets, Deployer displays the common properties but not source values.

5. In the bottom **Target Substitutions and Source Values** pane type the configuration values to substitute.
6. Click **Save Substitutions** to save the substitutions, or **Restore Defaults** to clear the changes you made.

Exporting and Importing Substitute Configuration Values

You can export and import substitute configuration values for both runtime-based and repository-based deployment projects.

To export and import substitute configuration values

1. Export the substitute configuration values from a deployment map as follows:
 - a. In the sourceDeployer, go to the **Deployer > Projects > project > Map** page.

- b. Click the deployment map that contains the substitute configuration values to export. Deployer displays the deployment map properties in the right-hand pane.
 - c. Click **Export Variable Substitution**. Deployer creates a file that contains the substitute configuration values for the assets in the project. The file is named *project_map.vs* and is stored in the *Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\outbound* directory. Deployer also allows you to save the file to your local file system.
 - d. If you exported substitute configuration values for scheduled tasks, open the *project_map.vs* file in an XML editor and set the task ID for each scheduled task to the task ID used on the target Integration Server.
2. Import the substitute configuration values into a deployment map as follows:
 - a. Copy the *project_map.vs* file to the *Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound* directory on the machine that hosts the target Deployer.
 - b. In the target Deployer, go to the **Projects > project > Map** page.
 - c. Click the deployment map into which to import the substitute configuration values. Deployer displays the deployment map properties in the right-hand pane.
 - d. Click **Import Variable Substitution**.
 - e. Select the *project_map.vs* file you just copied to the inbound directory.

If you receive the error message "Input XML map information is not valid" while importing a variable substitution .vs file, open the file and do the following:

 - a. Make sure the project contains all deployment sets specified on the DeploymentSet nodes.
 - b. For DeploymentSet nodes, make sure the PluginGroup value is set to either true or false, and the PluginType value is correct.
 - c. Make sure each DeploymentSet node is mapped to the correct TargetSystem name as specified in the exported *project_map.vs* file
 - d. Try to import again.

12 Deploying a Project

■ Overview	152
■ Preparing Integration Server to Stream Large Repository-Based Projects	152
■ Generating a Checkpoint	153
■ Deploying a Project	154
■ Post-Deployment Tasks	157
■ Rolling Back Target Servers	157

Overview

Deploying a project involves the following tasks:

- Prepare Deployer to stream large repository-based projects (optional).
- Generate a checkpoint. See ["Generating a Checkpoint" on page 153](#).
- Deploy or simulate the deployment of a project. See ["Deploying a Project" on page 154](#).
- Post-deployment tasks. See ["Post-Deployment Tasks" on page 157](#).
- Roll back the target servers. See ["Rolling Back Target Servers" on page 157](#).

Preparing Integration Server to Stream Large Repository-Based Projects

Note: For information about streaming large runtime-based projects, see the description of the **Large File Support** option in ["Setting General Deployment Defaults" on page 79](#).

If you choose to stream repository-based projects from Deployer to the target server, you must set certain server configuration settings on every target Integration Server hosting the runtime and Deployer. The build size of a project containing Integration Server packages and webMethods files can be up to 4GB.

You can stream assets from Deployer to target servers for the following runtime types:

- Integration Server
- Trading Networks
- BPM process models
- EDA
- Event Server

Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

To set server configuration parameters

1. For every target Integration Server hosting the runtime type and Deployer, open the Integration Server Administrator and go to the **Settings > Extended > Edit Extended Settings** page.

2. Type the following server configuration parameters and values in the box. For complete information about these server configuration parameters, see the *webMethods Integration Server Administrator's Guide*.
 - `watt.server.SOAP.MTOMStreaming.enable=true`
 - `watt.server.SOAP.MTOMStreaming.cachedFiles.location=directory_path`
 - `watt.server.SOAP.MTOMStreaming.threshold=number_of_bytes`
3. Click **Save Changes** and restart Integration Server.

Generating a Checkpoint

You generate a *checkpoint* for a project when you want the option of rolling back the target server to the state it was in prior to deploying your project. The checkpoint contains a copy of the assets on the target server that will be replaced by the assets in the deployment sets. You can set Deployer to generate checkpoints automatically or you can generate checkpoints manually for both runtime-based and repository-based projects.

Keep the following points in mind when working with checkpoints:

- If you take multiple checkpoints for a deployment candidate, only the latest is retained.
- The target servers must be available for the checkpoint generation to be successful.

Generating an Automatic Checkpoint

When Deployer generates checkpoints automatically, it does so as the first step of the deployment process when you deploy a project. You set Deployer to create automatic checkpoints as part of creating the project.

To set automatic checkpoint generation

- For runtime-based projects, you set the project to generate checkpoints automatically through the **Checkpoint Creation** parameter. For more information, see "[Setting Default Properties for All Projects](#)" on page 79.
- For repository-based projects, Deployer generates automatic checkpoints when you enable transactional deployment through the **Enable Transactional Deployment** parameter. For more information, see "[Creating a Project](#)" on page 85.

Generating a Checkpoint Manually

You can generate a checkpoint manually for both runtime-based and repository-based projects.

To generate a checkpoint manually

1. In the **Deployment Candidates** list, click  in the **Checkpoint** column. The checkpoint report appears in the right-hand pane in the **Deployment History** area.
2. Click  next to **Checkpoint** in the **Report Type** column to display the report. In the checkpoint report, the term **EXTRACT** is used for assets that exist on the target system and have been extracted to a backup. The term **MISSING** is used for assets that do not exist on the target system and will be deleted during a roll back.

The report is also available under the name `CheckpointReport_reportID.xml` in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\pub\projects\project_name\checkpoints\deployment_map\project_name Checkpoint\reports` folder, where `project_name` is the name of the project and `deployment_map` is the name of the deployment map.

Deploying a Project

When you deploy a project, Deployer deploys the assets in the project to the target servers.

You can simulate a deployment before you actually deploy. When you simulate a deployment, Deployer generates a *simulation report* that scans the target servers and alerts you to some potential problems before you deploy. You can address problems and re-generate the simulation reports until all problems are resolved. A simulation report contains information such as the following:

- Assets that will be suspended during deployment.
- Assets that will be enabled after deployment.
- Changes that will occur on the target servers, such as the assets that will be added or overwritten, and configuration values that will be substituted for Integration Server assets.
- Messages about problems, such as unresolved dependencies.

To deploy a project

1. If you chose to suspend triggers, ports, and scheduled tasks, but a service is triggered by one of these assets before Deployer suspends them, and the service is a long-running service, Deployer might overwrite the service during deployment. Make sure long-running services have completed.
2. In Deployer, go to the **Deployer > Projects** page.
3. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
4. In the **Name** column, click the project.

5. In the right-hand pane, click  **Deploy**. Deployer displays the **Projects > project > Deploy** page and lists all deployment candidates that exist for the selected project.
6. In the left-hand pane, click **Create Deployment Candidate**.
7. Set the **Create Deployment Candidate** parameters as follows:

Parameter	Description
Name	Accept the default deployment candidate name or replace it with a name that you choose. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Description	Type a description for the deployment candidate. The description length has no limit and can include any characters.
Project Build	(Run-time based deployment only.) Click the project build to deploy.
Deployment Map	Click the deployment map that identifies the target servers to which to deploy the assets. If the words Missing referenced assets appear next to the map name in the list, it means that you resolved an unresolved dependency using the Exists option, but the referenced asset does not exist on the target server. You can place the referenced asset on the target servers, or you can return to the project definition stage and re-resolve the dependency in a different way. For more information, see "Resolving Dependencies" on page 113 (for runtime-based deployment) or "Resolving Dependencies" on page 120 (for repository-based deployment). If you do not address the problem during the mapping task, Deployer will write a message about the problem to the simulation report. If you deploy without addressing the problem, Deployer will not deploy the deployment set.

8. Click **Create**.

In the candidate list in the left-hand pane, if the selected build and the current project definition are in sync, the **Status** column shows . If the project definition has changed since the build was created, the column shows .

9. If you want to see the progress report, click  in the **Progress Report** column.

The progress report displays the updates for simulate, deploy, checkpoint and rollback requests as they occur. This is useful in the case where the deployment build is large and it takes a long time to complete the action.

Note: If you are deploying a runtime-based project, you can rebuild the project build before proceeding. For instructions, see ["Rebuilding a Build" on page 139](#).

10. If you want to simulate the deployment, in the **Deployment Candidates** list, click  in the **Simulate** column.

The simulation report appears in the right-hand pane in the **Deployment History** area. Click  next to **Simulation** in the **Report Type** column to display the report. Read the report and address all problems. The report is also available under the name *project_name*_previewReport_reportID.xml in the *Integration Server_directory*\instances*instance_name*\packages\WmDeployer\pub\projects*project_name*\targets*deployment_map*\reports folder, where *project_name* is the name of the project and *deployment_map* is the name of the deployment map.

Note: If you do not address all problems at this time, you will probably experience errors during the deployment. For instructions on resolving unresolved dependencies, see ["Resolving Dependencies" on page 113](#) (for runtime-based deployment) or ["Resolving Dependencies" on page 120](#) (for repository-based deployment).

11. Click  in the **Deploy** column for the deployment candidate. Deployer does the following:

- Deploys the assets in the project to the target servers.
- Creates a deployment report and lists the report in the **Deployment History** area. Click  next to **Deployment Report** in the **Report Type** column to display the report. The report contains similar information to the simulation report, except that the events have actually occurred at this point. The report is also available under the name *project_name*_auditReport_reportID.xml in the *Integration Server_directory*\instances*instance_name*\packages\WmDeployer\pub\projects*project_name*\targets*deployment_map*\reports folder, where *project_name* is the name of the project and *deployment_map* is the name of the deployment map.
- If you are creating a runtime-based deployment project, Deployer performs the following additional tasks:
 - If you chose automatic checkpointing or automatic rollback in the project properties, Deployer automatically generates a checkpoint at this time. If you chose manual checkpointing and no checkpoint exists, Deployer asks whether you want to deploy anyway. If you deploy without a checkpoint, you will not be able to roll back the target servers.

- If the project build contains deletion set definitions, Deployer deletes the specified assets from the target servers you identified in the selected deployment map.
- If you are creating a repository-based deployment project and you set the **Enable Transactional Deployment** property to **Yes**, Deployer creates the checkpoint for the target server. For more information about the **Enable Transactional Deployment** property, see ["Creating a Project" on page 85](#).

Post-Deployment Tasks

- If you deployed JMS triggers, do the following:
 1. Create the same JMS alias connections on the target Integration Servers that exist on the source Integration Servers. Then reload the packages that contain the triggers.
 2. Enable the JMS triggers.
 3. Configure the queue or topic for each JMS trigger on the message provider for the target Integration Servers.

For instructions, see *Using webMethods Integration Server to Build a Client for JMS*.

- If you deployed My webMethods Server rules, the order in which the deployed rules are resolved with the existing rules on the target servers might need modification. Review the rule order and modify as necessary.
- If you deployed a process that uses e-forms with the project property **Enable process for execution** set to **No** (see ["Creating a Project" on page 85](#)), the e-form listener associated with the process cannot be enabled on the target server and therefore the process cannot be triggered on the target server. To make the process triggerable on the target server, enable it for execution and then enable the e-form listener.

Rolling Back Target Servers

If deployment to a target server fails and the target environment is in an inconsistent state, or a deployment is successful but the deployed assets are not working as expected, you can use Deployer's roll back feature to undo the deployment. When you roll back a deployment, Deployer rolls back the target server to the last checkpoint generated for the project. For more information about generating checkpoints, see ["Generating a Checkpoint" on page 153](#).

You can set Deployer to roll back target servers automatically or you can roll back target servers manually after deployment.

Rolling Back Target Servers Automatically

Deployer automatically rolls back target servers for runtime-based projects in these cases:

- You set the **Rollback on Error** project setting to **Automatic** (see ["Setting General Deployment Defaults" on page 79](#)). If the deployment fails on a target server, Deployer automatically rolls back that target server.
- Deployment failed to a target group whose **Rollback All on Failure** setting is **Yes** (see ["Creating Target Groups" on page 73](#)). If deployment to any server in such a target group fails, Deployer automatically rolls back all servers in the target group.

Deployer automatically rolls back the target servers for repository-based projects when transactional deployment is enabled for the project and deployment fails. For more information about enabling transactional deployment, see ["Creating a Project" on page 85](#).

Rolling Back Target Servers Manually

For runtime-based projects, you can roll back target servers manually at any time after deployment if you performed *both* of the following:

- You set the **Rollback on Error** project setting to **Manual**. For more information about this setting, see ["Setting Default Properties for All Projects" on page 79](#).
- You did not deploy to a target group whose **Rollback All on Failure** setting is **Yes**. For more information about this setting, see ["Creating Target Groups" on page 73](#).

For repository-based projects, you can roll back target servers manually if you performed *either* of the following:

- You enabled transactional deployment to create an automatic checkpoint through the **Enable Transactional Deployment** parameter. For more information about transactional deployment, see ["Creating a Project" on page 85](#).
- You generated a manual checkpoint for the project. For more information about generating a manual checkpoint, see ["Generating a Checkpoint Manually" on page 153](#).

To roll back target servers manually

1. In the **Deployment Candidates** list, click  in the **Rollback** column.

Deployer displays the rollback report in the right-hand pane in the **Deployment History** area.

2. To display the rollback report, click  next to **Rollback** in the **Report Type** column. The report is also available under the name `project_name_auditReport_reportID.xml` in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\pub\projects\project_name\targets\deployment_map\reports` folder,

where *project_name* is the name of the project and *deployment_map* is the name of the deployment map.

If you rolled back an IS & TN deployment set, the following apply:

- If the **Activate After Deployment** option for a package was set to **Inbound Only**, the report will warn that the package is not present on the target Integration Servers. You can ignore this warning.
- If the deployment set included webMethods files, the directory structure for those files remains in the webMethods installation directory on the target servers. You can delete the directories manually.
- If you deployed Trading Networks document attributes, field definitions, binary types, or profile security data, Deployer does not roll them back.

If you rolled back a ProcessModel deployment set, the rollback behavior varies, as follows:

- For process models you deployed that were versions of existing process models on the target servers, Deployer rolled back the deployed versions from the target servers.
- For deployed process models that were new on the target servers, Deployer disables the deployed process models but does not remove them from the target servers.

If you rolled back Universal Messaging assets, the rollback behavior is as follows:

- Deployer will not roll back the assets if the port is already being used by another existing interface.
- Deployer does not roll back nested security groups. For example, if group1 contains group2, the rollback will not restore the nested group2.

13 Using Deployer Commands

■ Overview	162
■ Installing Command Line Interface Only	162
■ Creating and Running Scripts	162
■ Specifying Log On Parameters	165
■ Error Handling and Logging	167
■ General and Project Commands	168
■ Build Commands	172
■ Commands for Repository-Based Deployment	177
■ Map Commands	178
■ Deployment Commands	182

Overview

You can use the command line interface to enter commands at a command prompt. The command line interface allows you to use many of the same features that are available in the Deployer GUI.

Installing Command Line Interface Only

You can install the Deployer command line interface on a machine, without installing the rest of Deployer or a host Integration Server. You might do this if you are using an automated deployment procedure that spans multiple machines.

To install only the Deployer command line interface on a machine, copy the files listed below from the indicated location on an existing remote Deployer host machine to the same location on the local machine.

File	Location
CLI.jar	<i>Integration Server_directory/packages/WmDeployer/lib</i>
log4j.properties, Deployer.{bat sh}	<i>Integration Server_directory/packages/WmDeployer/bin</i>
wm-isclient.jar	<i>Software AG_directory/common/lib</i>
jargs.jar, log4j.jar, gf.javax.mail.jar	<i>Software AG_directory/common/lib/ext</i> for jargs.jar and log4j.jar <i>Software AG_directory/common/lib/glassfish</i> for gf.javax.mail.jar

Edit the Deployer.{bat|sh} file you copied to the local machine to point to the jar files you copied and to a JDK or JRE 1.6 on the local machine.

Creating and Running Scripts

You can enter Deployer commands at a command prompt or you can create scripts that execute commands automatically. If you create a script, Deployer runs the commands in the order in which they appear in the script.

To invoke Deployer from the command line and execute a script, use the command for your operating system as follows:

For...	Command
Windows or UNIX	Deployer.{bat sh} <i>path_to_file</i>
Mac	deployerMac.sh <i>path_to_file</i>

You can also call scripts from other automated procedures, such as other scripts.

The sample script below automates these tasks on a Windows system:

- Imports a build that was exported from a test environment. Deployer automatically creates the deployment project and deployment sets.
- Displays the build contents on the console.
- Imports the deployment map.
- Imports substitute configuration values for Integration Server assets into the deployment map.
- Creates a deployment candidate.
- Generates a checkpoint, simulates the deployment, and deploys the build.

```
:environment
set host=%1
set port=%2
set user=%3
set pwd=%4
set project=testProject
set build=DemoBuild
set depCandidate=DemoDC
set depMap=DemoMap
rem ----clear the ERRORLEVEL system variable to avoid any side effects of
previous executions cases
set ERRORLEVEL=
:importBuild
set importB=%project%_ExportedBuild_%build%
IF "% ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Importing Build %ImportB%
ECHO -----
call Deployer.bat --import -buildFile %importB% -host %host% -port %port% -user
%user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set importB=
set nextAction=describeBuild
GOTO verifyStatus
:describeBuild
IF "%ERROR LEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Describing %build%
ECHO -----
call Deployer.bat --describe -build %build% -project %project% -host %host% -port
```

```

%port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set nextAction=buildit
GOTO verifyStatus
:importMap
set importM=%project% %depMap%.map
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Importing Map %ImportM%
ECHO -----
call Deployer.bat --import -mapFile %importM% -project %project% -host %host%
-port
%port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set importM=
set nextAction=importVarSub
GOTO verifyStatus
:importVarSub
set importV=%project% %depMap%.vs
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Importing Varsub %ImportV%
ECHO -----
call Deployer.bat --import -varsub -vsFile %importV% -map %depMap% -project
%project% -host %host% -port %port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set importV=
set nextAction=createDC
GOTO verifyStatus
:createDC
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Creating Deployment Candidate %depCandidate%
ECHO -----
call Deployer.bat --create -dc %depCandidate% -build %build% -map %depMap%
-project
%project% -host %host% -port %port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set nextAction=simulate
GOTO verifyStatus
:simulate
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Performaing deployment simulation on deployment candidate %depCandidate%
ECHO -----
call Deployer.bat -host %host% -port %port% -user %user% -pwd %pwd%
--simulate -project %project% -dc %depCandidate%
@echo off
echo.
echo.
echo.

```

```

set nextAction=checkpoint
GOTO verifyStatus
:checkpoint
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Performing CHECKPOINT operation of %depCandidate%
ECHO -----
echo %project%
echo %depCandidate%
call Deployer.bat --checkpoint -project %project% -dc %depCandidate% -host %host%
-port %port% -user %user% -pwd %pwd%
@echo off
echo .
echo .
echo .
set nextAction=deploy
GOTO verifyStatus
:deploy
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO DEPLOYING %depCandidate%
:VerifyStatus
IF "%ERRORLEVEL%" == "8" ECHO "<<<ERROR>>>"
IF "%ERRORLEVEL%" == "4" ECHO "<<<WARNING>>>"
IF "%ERRORLEVEL%" == "0" ECHO "<<<SUCCESS>>>"
echo.
echo.
goto %nextAction%
:FINISH
echo.
echo.
echo Completed.
set host=
set port=
set user=
set pwd=
set project=
set build=
set depCandidate=
set ERRORLEVEL=
@echo on

```

Specifying Log On Parameters

All Deployer commands require parameters for logging onto the Integration Server that hosts the Deployer. You can have Deployer commands connect to the Integration Server using HTTP or HTTPS.

If you want the Deployer commands to log on using HTTP, you can use an existing HTTP port on the Integration Server or configure a new one. If you want the Deployer commands to log on using HTTPS, you must do the following:

- Use an existing HTTPS port on the Integration Server or configure a new one.
- Place the command line interface's client certificate, private key, and signing authority's certificate on the Integration Server host machine.

- Map the command line interface's client certificate to an Integration Server user that has Administrator or Developer privileges.

For instructions on these tasks, see *webMethods Integration Server Administrator's Guide*.

When you run Deployer commands, the log on parameters you provide depend on whether you want to use HTTP or HTTPS, as follows:

- The log on parameters for logging onto an HTTP port are as follows:

```
Deployer.{sh|bat} --command -host host -port port -user user -pwd password
```

- The logon parameters for logging onto an HTTPS port are as follows:

```
Deployer.{sh|bat} --command -host host -port port -user user -pwd password
-useSSL -senderCert path_to_cert -privKey path_to_key -caCert path_to_cert
```

Parameter	Description
<code>-host host -port port</code>	Host machine and port for the Integration Server to log on to.
<code>-user user -pwd password</code>	User name and password to use to log on to the Integration Server. Note: If you do not provide a password, Deployer will prompt you for it.
<code>-useSSL</code>	Tells the Deployer command to log on to an HTTPS port.
<code>-senderCert path_to_cert</code>	Command line interface's client certificate.
<code>-privKey path_to_key</code>	Command line interface's private key.
<code>-caCert path_to_cert</code>	Command line interface's signing authority's certificate. Note: The certificates and private key do not exactly match the ones in the Integration Server installation.

Parameter	Description
	for the command line interface, the command will fail.

Creating a Configuration File for Log On Parameters

You can save time by creating a configuration file that specifies the values to use for the log on parameters and then pointing commands to the configuration file. Create the configuration file using a text editor and specify the appropriate parameter values, as specified above. For example:

```
host=idcauto1
port=5555
user=Administrator
pwd=1xcfdg55
host=idcauto1
port=5555
useSSL=true
senderCert=C:/files/SenderCert.der
privKey=C:/files/SenderPrivKey.der
caCert=C:/files/SenderCACert.der
```

Save the file with the extension `.cnf` and store it in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\bin` directory.

To point a command to the configuration file, specify the following on the command instead of the log on parameters:

```
Deployer.{sh|bat} --command -configfile file
```

Parameter	Description
<code>command</code>	Command to run.
<code>-configfile file</code>	Full path to the configuration file.

Error Handling and Logging

Deployer logs errors that occur during command line operations in the Deployer command line log file. The log file is named `CLI.log` and is located in a directory inside the current working directory. For example, if your working directory is `Integration Server_directory\instances\instance_name\packages\WmDeployer`, `CLI.log` is located in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\logs` directory.

Typical command line errors include required options that were not specified and invalid parameter values. Execution errors can include connectivity and authentication errors.

The maximum size for the CLI.log file is 100 KB. When it reaches the maximum size, it archives the log by renaming the file CLI.log.old and creating a new CLI.log file.

General and Project Commands

This section describes the commands to display Deployer usage information and product details and to maintain projects.

About

The `--about` command displays the following details about Deployer:

- JVM version number
- Publisher information
- Build number
- Package name
- Copyright information
- Integration Server version number

Run the following command to see the project details:

```
Deployer.{sh|bat} --about -host host -port port -user user_name -pwd password
```

Deleting a Project

Run the following command to delete a project:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --delete -project project  
-host host -port port -user user_name -pwd password
```

For more information about deleting projects, see ["Deleting a Project" on page 93](#).

Displaying Project Properties

Run the following command to display project properties:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --getProjectProperties -project project  
-host host -port port -user user_name -pwd password
```

Exporting Deletion Sets from a Project

When you run the `--export` command, Deployer creates a file that contains the definitions. The file is named `project_deleteSets.xml` and is stored in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\outbound` directory.

Run the following command to export a deletion set:

Note: You must have Define ACL authorization to run this command.

```
Deployer.[sh|bat] --export -deleteSpec -project project
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-project</code> <code>project</code>	Project whose deletion set definitions to export.
<code>-overwrite</code>	If the project already contains a file with the same name, this option tells Deployer to overwrite it. If you do not overwrite, and a file with the same name exists, Deployer issues an error and ends the command.

Importing Deletion Set Definitions into a Project

Before you can import deletion set definitions, you must copy the exported `project_deleteSets.xml` file to the `Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound` directory.

If the project already contains a deletion set with the same name as one you are importing, Deployer issues an error and ends the command.

Run the following command to import a deletion set into a project:

Note: You must have Define ACL authorization to run this command.

```
Deployer.{sh|bat} --import -deleteSpec definitions_file -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-deleteSpec</code> <code>definitions_file</code>	Full path to the file that contains the definitions to import. Definition files are named <code>project_deleteSets.xml</code> and are located in the <code>Integration Server_directory\instances</code>

Parameter	Description
	<code>\instance_name \packages \WmDeployer \replicate \inbound</code> directory.
<code>-project project</code>	Project into which to import the definitions.

Exporting Project Properties

When you export a project's properties, Deployer creates a file that contains the project property settings. The file is named `project.properties` and is stored in the `Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \outbound` directory. For more information about exporting project properties, see ["Exporting and Importing Project Properties" on page 89](#).

Run the following command to export project properties:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --export -projectProperties project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>projectProperties</code> <code>project</code>	Project from which to export properties.

Importing Project Properties

Importing properties into a project overwrites the existing properties for that project.

Before you can import project properties, you must copy the exported `project.properties` file to the `Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \inbound` directory on the machine that hosts the target Deployer.

You can edit the properties before you import them (see ["Editing a Deployment Map, Project Properties, or Substitute Configuration Values" on page 179](#)). If you do, keep in mind the following:

- You can specify ALWAYS or NEVER for the `overwrite` property.
- You can specify REPLACE OR MERGE for the `deployTNRules` property.
- You can specify true, false, or selected for the `stopTriggers` property.
- You can specify true or false for all other properties.

If you specify a value for a property that is not allowed, Deployer resets the property to the default value when it imports the project properties.

Run the following command to import a project's properties:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --setProjectProperties -project project
-projectFile properties_file
-host host -port port -user user_name -pwd password
```

Parameter	Description
-project <i>project</i>	Project into which to import the properties.
-projectFile <i>properties_file</i>	Full path to the file that contains the properties to import. These files are named <i>project.properties</i> and are located in the <i>Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound</i> directory.

Help

Run the following command to view a list of Deployer commands you can use in the command line interface:

```
Deployer.{sh|bat} --help -command command_string
```

Parameter	Description
-command <i>command_string</i>	Command for which you want usage information.

Listing Builds, Maps, or Deployment Candidates for a Project

Run the following command to list builds, maps, or deployment candidates for a build:

Note: You must have the correct authorizations to run this command depending on whether you want to list builds, maps, or deployment candidates.

- To list builds, you must have Build ACL authorization.
- To list maps, you must have Map ACL authorization.
- To list deployment candidates, you must have Deploy ACL authorization.

```
Deployer.{sh|bat} --list -candidate {Build|Map|DC} -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-candidate {Build Map DC}</code>	Whether to list builds, maps, or deployment candidates.
<code>-project <i>project</i></code>	Project that contains the builds, maps, or deployment candidates to list.

Locking Projects

Run the following command to lock a project:

```
Deployer.{sh|bat} --lockProject -project project
-host host -port port -user user_name -pwd password
```

Unlocking Projects

Run the following command to unlock a project:

```
Deployer.{sh|bat} --unlockProject -project project
-host host -port port -user user_name -pwd password
```

Build Commands

This section describes the commands to create, export, import, and display details about a build.

Creating a Project Build

When creating a project build, the build creation will fail if there are any unresolved dependencies. For instructions on resolving unresolved dependencies, see ["Resolving Dependencies" on page 113](#).

Run the following command to create a project build:

Note: You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --create -build build -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
<code>-build <i>build</i></code>	Name of the build to create. The build name can be up to 32 characters long and can include any

Parameter	Description
	characters that are valid for a file name in your operating system.
<code>-project <i>project</i></code>	Project from which to create the build.
<code>-reportFilePath <i>report_path</i></code>	Full path to the local directory where Deployer stores the generated build report.

Listing Builds for a Project

Run the following command to list the builds in a project:

Note: You must have View ACL authorization to run this command.

```
Deployer.{sh|bat} --list -candidate build -project project
-host host -port port -user user_name -pwd password
```

Displaying Contents of a Build

Run the following command to display the contents of a specific build:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -build build -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-build <i>build</i></code>	Build whose contents to display.
<code>-project <i>project</i></code>	Project to which the build belongs.

Displaying Substitute Configuration Values for Integration Server Assets in a Build

Use the following command to display the substitute configuration values for Integration Server assets in a build:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -build build -project project -varsub
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-build <i>build</i></code>	Build whose substitute configuration values to display.
<code>-project <i>project</i></code>	Project to which the build belongs.
<code>-varsub</code>	Displays the substitute configuration values.

Displaying Contents of a Build File

Use the following command to display the contents of a build file:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -buildFile build_file -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-buildFile <i>build_file</i></code>	Full path to the build file whose contents to display. Build files are named <i>project_build</i> and are located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\outbound directory.
<code>-project <i>project</i></code>	Project to which the build belongs.

Displaying Substitute Configuration Values for Integration Server Assets in a Build File

Use the following command to display the substitute configuration values for Integration Server assets in a build file:

Note: You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -buildFile build_file -project project -varsub
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-buildFile <i>build_file</i></code>	Full path to the build file whose substitute configuration values to display. Build files are named <i>project_build</i> and are located in

Parameter	Description
	the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\outbound directory.
-project <i>project</i>	Project to which the build belongs.
-varsub	Displays the substitute configuration values.

Exporting a Build from a Project

Use the following command to export a build from a project:

Note: You must have Build ACL authorization to run this command.

```
Deployer.[sh|bat] --export -build build -project project
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
-build <i>build</i>	Build to export.
-project <i>project</i>	Project to which the build belongs.
-overwrite	If the project already contains a build with the same name, this options tells Deployer to overwrite it. If you do not overwrite, and a build with the same name exists, Deployer issues an error and ends the command.

Deployer creates a file that contains the build. The file is named *project_build* and is stored in the *Integration Server_directory*\instances*instance_name* \packages\WmDeployer\replicate\outbound directory.

Importing a Build File into a Project

Before you can import a build, you must copy the exported *project_build* file to the *Integration Server_directory*\instances*instance_name* \packages\WmDeployer\replicate\inbound directory on the machine that hosts the target Deployer.

Run the following command to import a build file into a project.

Note: You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --import -buildFile build_file -project project
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-buildFile</code> <code>build_file</code>	Full path to the build file that contains the deployment map to import. Build files are named <code>project_build</code> and are located in the <code>Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound</code> directory.
<code>-project project</code>	Project into which to import the build.
<code>-overwrite</code>	If the project already contains a build with the same name, this options tells Deployer to overwrite it. If you do not overwrite, and a build with the same name exists, Deployer issues an error and ends the command.

Listing Build Reports

Run the following command to list the build reports for a project.

Note: You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --list -candidate buildReport -build build -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-build build</code>	Build for which to list build reports.
<code>-project</code> <code>project</code>	Project to which the build belongs.

Displaying a Build Report

Run the following command to display a build report.

Note: You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --showReport -candidate buildReport -build build
-id integerId -project project -host host -port port -user user_name
-pwd password
```

Parameter	Description
<code>{-build <i>build</i></code>	Build whose build report to display.
<code>id <i>report_</i> <i>identifier</i></code>	Identifier for the report to display. Use the <code>--list</code> command (see "Listing Build Reports" on page 176) to display report identifiers, as well as the date and time each report was generated.
<code>-project <i>project</i></code>	Project to which the build belongs.

Commands for Repository-Based Deployment

This section describes commands you can run specific to building indexes for repository-based deployment.

Rebuilding the Index with the Build Script

If the index you created becomes corrupted or the repository index is accidentally deleted from the repository, you can use the `createIndex` command to recreate the index.

By default, the `createIndex` command rebuilds the index in the location specified by the `build.output.dir` property you specified in ["Setting the Properties for the Build" on page 32](#). You can override the default repository path by specifying the path of the repository with the `-Drepo.dir` command.

Note: When you follow this procedure to rebuild the index, the build script creates *only* the index. To build the index, check out the asset sources, version the assets, and build the composites and descriptors in the repository, you must run the build script as described in ["Running the Build Script and Rebuilding the Index" on page 41](#).

Run one of the following commands from the `Software AG_directory\common\AssetBuildEnvironment\bin` directory:

For this platform...

Run the following command...

Windows

```
build.bat -
Drepo.dir=repository_pathcreateIndex
```

For this platform...**Run the following command...**

UNIX

```
build.sh -Drepo.dir=
repository_pathcreateIndex
```

Where *repository_path* is the full path of the repository directory.

Note: If you do not specify a path for `-Drepo.dir`, the build script indexes the repository specified by the `build.output.dir` property. For more information about the `build.output.dir` property, see ["Setting the Properties for the Build" on page 32](#).

Map Commands

This section describes the commands to list, import, export, edit, and delete deployment maps.

Note: You must have Map ACL authorization to run the commands in this section.

Listing All Deployment Maps

Run the following command to list the deployment maps for a candidate.

```
Deployer.{sh|bat} --list -candidate mapFile
-host host -port port -user user_name -pwd password
```

Exporting a Deployment Map from a Project

When you export a deployment map from a project, Deployer creates a file that contains the deployment map. The file is named *project_map*.map and is stored in the *Integration Server_directory*\instances*instance_name* \packages\WmDeployer\replicate \outbound directory.

Run the following command to export a deployment map:

```
Deployer.{sh|bat} --export -map map -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-map map</code>	Deployment map to export.
<code>-project project</code>	Project to which the map belongs.

Editing a Deployment Map, Project Properties, or Substitute Configuration Values

After you export a deployment map or substitute configuration values, you can edit the resulting file before importing it into the other environment. For example, if you want to map a deployment set to a different target server, you could change the `targetServer alias` attribute to reflect the new target server name.

You can open a deployment map or substitute configuration value file using any XML editor. A deployment map file has the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeploymentMap description="description of map" mapName="mapSetName"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeploymentSets>
    <DeploymentSet name="deploymentSetName" pluginType="pluginType">
      <targetGroups>
        <targetGroup alias="targetGroupName"/>
      </targetGroups>
      <targetServers>
        <targetServer alias="targetServerAlias"/>
      </targetServers>
    </DeploymentSet>
  </DeploymentSets>
</DeploymentMap>
```

To specify an additional target server, target group, or deployment set in the same deployment map, repeat the attribute for each addition. For example, a deployment set that is mapped to multiple target servers is defined as follows:

```
<DeploymentMap>
  <DeploymentSets>
    <DeploymentSet name="deploymentsetA" pluginType="MWS">
      <targetGroups>
        <targetGroup alias="<targetGroupName>"/>
      </targetGroups>
      <targetServers>
        <targetServer alias="server1"/>
        <targetServer alias="server2"/>
        <targetServer alias="server3"/>
      </targetServers>
    </DeploymentSet>
  </DeploymentSets>
</DeploymentMap>
```

Importing a Deployment Map Into a Project

Before you can import a deployment map, you must copy the exported `project_map.map` file to the `Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound` directory on the machine that hosts the target Deployer. You can edit the map before you import it (see ["Editing a Deployment Map, Project Properties, or Substitute Configuration Values" on page 179](#)).

Run the following command to import a deployment map into a project:

```
Deployer.{sh|bat} --import -mapFile map_file -project project
```

```
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-mapFile</code> <code><i>map_file</i></code>	Full path to the map file that contains the deployment map to import. Map files are named <i>project_map</i> .map and are located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\inbound directory.
<code>-project</code> <code><i>project</i></code>	Project into which to import the map.
<code>-overwrite</code>	If the project already contains a map with the same name, this options tells Deployer to overwrite it. If you do not overwrite, and a map with the same name exists, Deployer issues an error and ends the command.

Exporting Substitute Configuration Values for Integration Server Assets from a Deployment Map

Run the following command to substitute configuration values for Integration Server assets from a deployment map:

```
Deployer.{sh|bat} --export -map map -project project -varsub  
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-map <i>map</i></code>	Deployment map from which to export substitute configuration values.
<code>-project</code> <code><i>project</i></code>	Project to which the map belongs.
<code>-varsub</code>	Exports the substitute configuration values.

Deployer creates a file that contains the substitute configuration values. The file is named *project_map*.vs and is stored in the *Integration Server_directory*\instances*instance_name* \packages\WmDeployer\ replicate\outbound directory.

If you exported substitute configuration values for scheduled tasks, open the *project_map*.vs file in an XML editor and set the task ID for each scheduled task to the task ID used on the target Integration Server.

Note: If no substitute configuration values are specified in the deployment map, the Deployer creates a file with the complete structure but does not export any values.

Importing Substitute Configuration Variables for Integration Server Assets into a Deployment Map

Before you can import substitute configuration values into a deployment map, you must copy the exported *project_map.vs* file to the *Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound* directory on the machine that hosts the target Deployer.

You can open the *project_map.vs* file in an XML editor and edit the values before importing. For example, if you exported substitute configuration values for scheduled tasks, you must edit the file for each target Integration Server so that the task ID for each scheduled task is set to the task ID used on the target Integration Server.

Run the following command to import substituted configuration variables for Integration Server assets into a deployment map:

```
Deployer.{sh|bat} --import -varsub -vsFile project_map.vs -map map
-project project -validate {true|false}
-host host -port port -user user_name -pwd password
```

Parameter	Description
-varsub	Imports the variable substitution values.
-vsFile <i>project_map.vs</i>	File that contains the substitute configuration values to import. These files are named <i>project_map.vs</i> and are located in the following directory: <i>Integration Server_directory\instances\instance_name\packages\WmDeployer\replicate\inbound</i>
-map <i>map</i>	Deployment map into which to import the values.
-project <i>project</i>	Project that contains the map into which to import the values.
-validate {true false}	Whether Deployer should check the values to make sure they are valid for the target servers. If you specify true (validate), Deployer lists any servers that are not running on the console.

Deleting a Deployment Map from a Project

Run the following command to delete a deployment map from a project:

```
Deployer.{sh|bat} --delete -map map -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-map <i>map</i>	Deployment map to delete.
-project <i>project</i>	Project that contains the map to delete.

Deployment Commands

This section describes the commands to create, display information about, deploy, and delete deployment candidates and to generate checkpoints, simulate a deployment, roll back a target server, and generate reports.

Note: You must have Deploy ACL authorization to run the commands in this section.

Creating a Deployment Candidate

Run the following command to create a deployment candidate:

```
Deployer.{sh|bat} --create -dc deployment_candidate -build build -map map
-project project -host host -port port -user user_name -pwd password
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate to create.
-build <i>build</i>	Project build to use in the deployment candidate.
-map <i>map</i>	Deployment map to use in the deployment candidate.
-project <i>project</i>	Project to which the build and map belong.

Displaying Information About a Deployment Candidate

Run the following command to display information about a deployment candidate:

```
Deployer.{sh|bat} --describe -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate for which to obtain information, such as: <ul style="list-style-type: none"> ■ Name of the build and deployment map in the candidate. ■ Date the candidate was created. ■ All existing deployment reports for the candidate.
-project <i>project</i>	Project to which the deployment candidate belongs.

Deleting a Deployment Candidate

Run the following command to delete a deployment candidate:

```
Deployer.{sh|bat} --delete -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate to delete.
-project <i>project</i>	Project to which the deployment candidate belongs.

Generating a Checkpoint

Note: The target servers must be available for the checkpoint generation to be successful. For more information about checkpoints, see ["Checkpoint and Roll Back" on page 20](#).

Run the following command to generate a checkpoint:

```
Deployer.{sh|bat} --checkpoint -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate you plan to deploy.
<code>-project project</code>	Project to which the deployment candidate belongs.
<code>-reportFilePath report_path</code>	Full path to the local directory where Deployer stores the generated checkpoint report.

Simulating a Deployment

When you run this command and simulate a deployment, Deployer generates a simulation report. Display the simulation report as instructed in ["Displaying a Simulation, Rollback, or Deployment Report" on page 186](#) and address all problems.

Note: If you do not address all problems at this time, you will probably experience errors during deployment.

Run the following command to simulate a deployment:

```
Deployer.{sh|bat} --simulate -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate for which to simulate a deployment.
<code>-project project</code>	Project to which the deployment candidate belongs.
<code>-reportFilePath report_path</code>	Full path to the local directory where Deployer stores the generated simulation report.

Deploying

When you run this command, Deployer deploys the assets in the candidate's project build to the target servers in the candidate's deployment map. In addition, Deployer generates a deployment report. Display the deployment report as instructed in ["Displaying a Simulation, Rollback, or Deployment Report" on page 186](#).

Run the following command to deploy a deployment candidate:

```
Deployer.{sh|bat} --deploy -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -force
-reportFilePath report_path
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate to deploy.
-project <i>project</i>	Project to which the deployment candidate belongs.
-force	If no checkpoint exists for the deployment candidate (for example, because you chose to generate checkpoints manually, but did not do so), Deployer will not deploy unless you specify this parameter.
-reportFilePath <i>report_path</i>	Full path to the local directory where Deployer stores the generated deployment report.

Note: If you deploy without a checkpoint, you will not be able to roll back target servers.

Rolling Back Target Servers

When you roll back target servers, Deployer generates a rollback report. For information about displaying the rollback report, see ["Displaying a Simulation, Rollback, or Deployment Report" on page 186](#).

Run the following command to roll back target servers:

```
Deployer.{sh|bat} --rollback -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate whose deployed assets to remove from the target servers.
-project <i>project</i>	Project to which the deployment candidate belongs.
-reportFilePath <i>report_path</i>	Full path to the local directory where Deployer stores the generated rollback report.

Listing Simulation, Rollback, and Deployment Reports

Run the following command to list simulation, rollback, and deployment reports for a deployment candidate:

```
Deployer.{sh|bat} --list -candidate deploymentReport -dc deployment_candidate
-project project -host host -port port -user user_name -pwd password
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate whose simulation, deployment, and rollback reports to list.
-project <i>project</i>	Project to which the deployment candidate belongs.

Displaying a Simulation, Rollback, or Deployment Report

Run the following command to display a simulation, rollback, or deployment report for a deployment candidate:

```
Deployer.{sh|bat} --showReport -candidate deploymentReport
-dc deployment_candidate -id integerId -project project
-host host -port port -user user_name -pwd password -
```

Parameter	Description
-dc <i>deployment_candidate</i>	Deployment candidate whose simulation, deployment, or rollback report to display.
id <i>report_identifier</i>	Identifier for the report to display. Use the <code>--list</code> command (see " Listing Simulation, Rollback, and Deployment Reports " on page 186) to display report identifiers, as well as the date and time each report was generated.
-project <i>project</i>	Project to which the deployment candidate belongs.

14 Automating Project Creation

■ Overview	188
■ Exporting Projects for Use in Project Automator	188
■ Using Handles Instead of Passwords	189
■ Error Handling and Logging	191
■ Root Tag	191
■ Identifying Deployer	192
■ Setting Up Aliases for Source and Target Servers	192
■ Creating Projects	220
■ Running Project Automator	234

Overview

To configure automatic projects, you provide the necessary specifications for automated project creation in an XML file. This chapter describes the tags you can specify in the file. Only the root tag and the tag that identifies Deployer are required in the XML file.

Sample XML files are provided in the *Integration Server_directory/instances/instance_name/packages/WmDeployer/config* directory. There are two files: *ProjectAutomatorSampleForRepository.xml* provides an example of a repository-based automated project, and *ProjectAutomatorSampleForRuntime.xml* shows a sample runtime-based automated project. You can also export a project you created in the GUI for use in Project Automator. For more information about exporting a project from the GUI, see "[Exporting Projects for Use in Project Automator](#)" on page 188.

For complete contextual information about the features that each tag relates to, see the GUI chapters in this guide.

Exporting Projects for Use in Project Automator

After you create a project in the GUI, you can export the project to a specification XML file that you can then use to automate your project. You specify the data to include in the specification XML file. You can export the alias, deployment set, build, map, and deployment candidate definitions associated with the project.

Exporting projects from the GUI

1. Go to the **Tools > Export to Project Automator** page.
2. Complete the fields as follows:

Field	Entry
Project	Select the project to export.
Export Alias Definition	Optional. Click to export all of the alias definitions for the source and targets associated with the project.
Export Deployment and Deletion Set Definition	Optional. Click to export all of the deployment and deletion set definitions associated with the project. Exported deployment and deletion sets include the definition set and all of the associated assets.
Export Build Definition	Optional. Click to export all of the build definitions associated with the project.

Field	Entry
Export Map Definition	Optional. Click to export all of the map definitions associated with the project. The map definition includes all target servers, target groups, and clusters that are part of the deployment map.
Export Deployment Candidate Definition	Optional. Click to export all of the deployment candidate definitions associated with the project.

- Click **Export to Project Automator**.

Deployer exports the project specification XML file to the following location:

```
Integration Server_directory/instances/instance_name/packages/WmDeployer/replicate/outbound/projectName_ProjectAutomator.xml
```

Where *projectName* is the name of the project.

Using Handles Instead of Passwords

Project Automator uses the `pwd` attribute to store server passwords in projects. This attribute is encrypted the first time you run Project Automator. In order to avoid passing passwords in clear text the first time you run Project Automator, you can use a *password handle*. Password handles allow you to create a password on the host Integration Server along with a corresponding key (or handle) which you then store in clear text in the `pwdHandle` attribute. The handle is encrypted as an outbound password using the Password-Based Encryption (PBE) technology installed with Integration Server. For more information about how Integration Server manages outbound passwords, see *webMethods Integration Server Administrator's Guide*.

Project Automator gets the password associated to the password handle specified in the `pwdHandle` element. You create and manage password handles in the Deployer GUI. You can also delete and modify password handles as needed.

Keep the following points in mind when using password handles:

- Password handles are valid only when Project Automator is running on the same host Integration Server on which the password handles are created.
- When using password handles, Project Automator can connect only to a Deployer installed in same directory as Project Automator itself.

Creating Password Handles

Perform the following steps to create password handles.

To create password handles

1. From the Deployer GUI running on the same server as Project Automator, click **Deployer > Password Store**
2. Click **Create Password Store Entry**.
3. In the right-hand pane, under **Create Password Store Entry**, complete the following fields:

Field	Entry
Password Handle	The name of the password handle. This is the value you will specify in the <code>pwdHandle</code> attribute in Project Automator. Password handles cannot contain the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Password	The password to associate with the password handle.

4. Click **Create**.

Modifying Password Handle Associations

Perform the following steps to modify the password associated with a password handle.

To modify the password associated with the password handle

1. Click **Deployer > Password Store**.
2. Click the password handle you want to modify from the **Password Store Entries** list.
3. In the right-hand pane, in the **New Password** field, enter the new password to associate with the password handle.
4. Click **Update**.

Deleting Password Handles

Perform the following steps to delete password handles.

To delete password handles

1. Click **Deployer > Password Store**
2. Click  in the **Delete** column for the password handle.
Deployer displays a confirmation dialog.

3. Click **OK** to confirm that you want to delete the password handle.

Error Handling and Logging

Project Automator produces a log file (ProjectAutomatorReport.xml) that is controlled by a log4j property file stored in the *Integration Server_directory/instances/instance_name/packages/WmDeployer/bin* directory. You can change the properties.

```
<Report>
  <Messages type="info">
    <message>message
text</message>
    <message>message
text</message>
  </Messages type="info">
  <Messages type="error">
    <message category="category" errorCode="code" deploymentSet="set name "
    deploymentProject="project
name">message text</message>
  </Messages type="error">
</Report>
```

Below is an example of an error message:

```
<message category="projectError" errorCode="-41" deploymentSet="myDeploymentSet"
deploymentProject="TestProject">Error adding ACLs TestACL1, TestACL2 to
Deployment Set for project TestProject</message>
```

For error messages, you can write a program to parse the attribute values and take specified actions.

Root Tag

The root tag for Project Automator consists of the `<DeployerSpec>` tag and the `exitOnError` attribute, as follows:

```
<DeployerSpec exitOnError="true or false"></DeployerSpec>
```

The following table describes the attribute you can specify in the `<DeployerSpec>` tag.

Attribute	Description
<code>exitOnError</code>	Optional. Indicates how Project Automator should handle errors. Set to: <ul style="list-style-type: none"> ■ <code>true</code> to set Project Automator to report the error and terminate the first time it encounters an error. ■ <code>false</code> to set Project Automator to report errors as they occur, but continue processing. This is the default.

Identifying Deployer

You identify the Deployer on which you will perform the project tasks in the <DeployerServer> tag. The <DeployerServer> tag enables you to specify the values required to log on to the Integration Server that hosts the Deployer.

The following example shows how to use this tag:

```
<DeployerServer>
  <host>Integration
  Server host name or IP address :port </host>
  <user>user
  name </user>
  <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
</DeployerServer>
```

The following table describes the attributes you can specify in the <DeployerServer> tag.

Attribute	Description
host	Host name or IP address of the server.
user	User name of the server.
pwd	Password of the server. You must specify either pwd or pwdHandle. Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.
pwdHandle	The password handle. You must specify either pwd or pwdHandle. For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .

Setting Up Aliases for Source and Target Servers

You set up aliases for source and target servers, target groups, and source repositories in the <Environment> tag. For example:

```
<Environment>
  <{webMethods
  Broker|ProcessModel|IS|MWS|Optimize|EventServer|RulesServer|EDA|
  UniversalMessaging|TargetGroup|Repository}>
  <{broker|pm|is|mws|optimize|eventserver|rullesserver|edaserver|
  universalmessaging|rep}alias>
```

```

    tags
  </{broker|pm|is|mws|optimize|eventserver|rulesserver|edaserver|
  universal messaging|rep}alias>
  </{webMethods Broker|ProcessModel|IS|MWS|Optimize|EventServer|RulesServer|EDA|
  UniversalMessaging|TargetGroup|Repository}>
</Environment>

```

The sections below describe each tag within the <Environment> tag in detail.

If Deployer already contains an alias with the same name as one you define, Deployer overwrites the alias.

Note: The credentials for <user>, <pwd>, and <pwdHandle> asset tags must be those for a user with Administrator ACL authorization or for a user that belongs to a group that has Internal, Developer, and DeployerAdmin ACLs to create Deployer runtime aliases and projects.

Setting Up Aliases for Source Repositories

For repository-based deployment, you define the repository as the source server. This location identifies the repository directory from which the assets should be deployed.

Note: You can set up aliases for source repositories for repository-based deployment only.

```

<Repository>
  <repalias name="name">
    <type>FlatFile</type>
    <urlOrDirectory>directory_location</urlOrDirectory>
    <Test>true or false</Test>
  </repalias>
</Repository>

```

For more information about the values to supply for the following attributes, see ["Connecting to a Repository for Repository-Based Deployment" on page 72](#).

Attribute	Description
repalias name	The name to use for the repository alias. This attribute corresponds to the Name field.
type	The type of repository file. Set to FlatFile.
urlOrDirectory	The full path of the repository directory in which the composites are located. This attribute corresponds to the File Directory field.
test	Whether Deployer should test the connection to the source repository. Set to: <ul style="list-style-type: none"> ■ true to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of

Attribute	Description
	<p>the <DeployerSpec> tag. For more information, see "Root Tag" on page 191.</p> <ul style="list-style-type: none"> ■ <code>false</code> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Application Platform Deployment Endpoints

The following example illustrates how to set up aliases for target Application Platform deployment endpoints.

```
<ApplicationPlatform>
  <applicationplatformalias name="APP_Target ">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </applicationplatformalias>
</ApplicationPlatform>
```

For information on the values to supply for the tags below, see ["Connecting to Application Platform Servers" on page 58](#).

Attribute	Description
applicationplatformalias name	Name to assign to the Application Platform deployment endpoint. This attribute corresponds to the Name field.
host	Host name or IP address of the Software AG Platform Manager runtime. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> .

Attribute	Description
	<p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p>
<code>pwdHandle</code>	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
<code>useSSL</code>	<p>Whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <code>true</code> to use SSL to connect to the server. ■ <code>false</code> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
<code>version</code>	Version of the server. This attribute corresponds to the Version field.
<code>Test</code>	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <code>true</code> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <code>false</code> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Source and Target webMethods Brokers

You can set up aliases for source and target Broker Servers for either basic authentication, SSL authentication, or neither.

Basic Authentication

The following example illustrates how to set up a Broker alias that uses basic authentication.

```
<Broker>
  <brokeralias name="alias
name ">
  <brokerName>Broker
name </brokerName>
```

```

<clientGroup>client
group</clientGroup>
<host>Broker
server host</host>
<port>Broker
Server port</port>
<useBasicAuth>true</useBasicAuth>
  <user>basic
authorization user name</user>
  <pwd>basic
authorization password</pwd> OR <pwdHandle>handle</pwdHandle>
  <version>version_number</version>
<context>JNDI
context</context>
<Test>true/false</Test>
</brokeralias>
</Broker>

```

For detailed information on the values to supply for the following attributes, see ["Connect to Broker Servers" on page 63](#).

Attribute	Description
brokeralias name	Name to assign to the Broker Server. This attribute corresponds to the Name field.
brokerName	Name of the source or target webMethods Broker. This attribute corresponds to the BrokerName field.
clientGroup	Client group Deployer should use to access the source or target Broker Server. For target Broker Servers, specify <i>admin</i> . This attribute corresponds with the Client Group field.
host	Host name or IP address of the Broker Server. This attribute corresponds to the Host field.
port	Port for the Broker Server. This attribute corresponds to the Port field.
useBasicAuth	Set to <i>true</i> to connect to the Broker Server using basic authentication. This attribute corresponds to the Client Authentication > Username/Password check box.
user	Basic authentication user name. This attribute corresponds to the Username field.
pwd	Basic authentication password. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> .

Attribute	Description
	<p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
version	Version of the Broker Server. This attribute corresponds to the Version field.
context	JNDI context. Required if the Broker Server serves as a JNDI provider. This attribute corresponds to the Context field.
Test	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

SSL Authentication

The following example illustrates how to set up a Broker alias that uses SSL authentication.

```
<Broker>
  <brokeralias name="alias name">
    <brokerName>Broker name</brokerName>
    <clientGroup>client group</clientGroup>
    <host>Broker server host</host>
    <port>Broker Server port</port>
    <useSSL>>true</useSSL>
    <version>version_number</version>
    <keyStoreType>Deployer keystore type</keyStoreType>
    <keyStorePath>Deployer keystore path</keyStorePath>
    <keyStorepassword>Deployer keystore password</keyStorepassword>
    <trustStoreType>Deployer trust store type</trustStoreType>
    <trustStorePath>Deployer truststore path</trustStorePath>
    <context>JNDI context</context>
    <Test>true/false</Test>
  </brokeralias>
</Broker>
```

For detailed information on the values to supply for the following attributes, see ["Connect to Broker Servers" on page 63](#).

Attribute	Description
brokeralias name	Name to assign to the Broker Server. This attribute corresponds to the Name field.
brokerName	Name of the source or target webMethods Broker. This attribute corresponds to the BrokerName field.
clientGroup	Client group Deployer should use to access the source or target Broker Server. For target Broker Servers, specify <i>admin</i> . This attribute corresponds with the Client Group field.
host	Host name or IP address of the Broker Server. This attribute corresponds to the Host field.
port	Port for the Broker Server. This attribute corresponds to the Port field.
useSSL	Set to <i>true</i> to connect to the Broker Server using SSL authentication. This attribute corresponds to the Client Authentication > SSL check box.
version	Version of the Broker Server. This attribute corresponds to the Version field.
keyStoreType	File type of Deployer's keystore file. This attribute corresponds to the Keystore Type field.
keyStorePath	Full path to Deployer's keystore file. This attribute corresponds to the DeployerKeystore field.
keyStore password	Password that Deployer uses to access its keystore file. This attribute corresponds to the Keystore Password field.
trustStoreType	File type of Deployer's truststore file. This attribute corresponds to the Truststore Type field.
trustStorePath	Full path to Deployer's truststore file. This attribute corresponds to the DeployerTruststore field.

Attribute	Description
context	JNDI context. Required if the Broker Server serves as a JNDI provider. This attribute corresponds to the Context field.
Test	Whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

No Authentication

The following example illustrates how to set up a Broker alias that does not use client authentication.

```
<Broker>
  <brokeralias name="alias
name ">
    <brokerName>Broker
name</brokerName>
    <clientGroup>client
group</clientGroup>
    <host>Broker
server host</host>
    <port>Broker
Server port</port>
    <useSSL>false</useSSL>
    <version>version
number</version>
    <context>JNDI
context</context>
    <Test>true/false</Test>
  </brokeralias>
</Broker>
```

For detailed information about the values to supply for the following attributes, see ["Connect to Broker Servers" on page 63](#).

Attribute	Description
brokeralias name	Name to assign to the Broker Server. This attribute corresponds to the Name field.
brokerName	Name of the source or target webMethods Broker. This attribute corresponds to the BrokerName field.

Attribute	Description
clientGroup	Client group Deployer should use to access the source or target Broker Server. For target Broker Servers, type <i>admin</i> . This attribute corresponds to the Client Group field.
host	Host name or IP address of the Broker Server. This attribute corresponds to the Host field.
port	Port for the Broker Server. This attribute corresponds to the Port field.
useSSL	Set to <i>false</i> to connect to the Broker Server without any client authentication. This attribute corresponds to the Client Authentication > None check box.
version	Version of the Broker Server. This attribute corresponds to the Version field.
context	JNDI context. Required if the Broker Server serves as a JNDI provider. This attribute corresponds to the Context field.
Test	Whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Source and Target Process Model Servers

The following example illustrates how to set up aliases for source and target process model servers.

```
<ProcessModel>
  <pmalias name="alias name">
    <host>ProcessModel Server host</host>
    <port>ProcessModel Server port</port>
    <user>user name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
```

```
</palias>
</ProcessModel>
```

For detailed information about the values to supply for the following attributes, see ["Connect to BPM Process Model Servers" on page 65](#).

Attribute	Description
palias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
useSSL	Whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. This attribute corresponds to the Use SSL field.
version	Version of the server. This attribute corresponds to the Version field.
Test	Whether Deployer should test the connection to the servers. Set to:

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Source and Target Integration Servers

The following example illustrates how to set up aliases for source and target Integration Servers.

```
<IS>
  <isalias name="alias
name">>
    <host>Integration
Server host</host>
    <port>Integration
Server port</port>
    <user>user
name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <installDeployerResource>true/false</installDeployerResource>
    <Test>true/false</Test>
    <executeACL>acl</executeACL>
  </isalias>
</IS>
```

For information on the values to supply for the tags below, see ["Connecting to Integration Servers and Trading Networks Servers" on page 56](#).

Attribute	Description
<code>isalias name</code>	Name to assign to the server.
<code>host</code>	Host name or IP address of the server.
<code>port</code>	Port for the server.
<code>user</code>	Optional. User name for a user account with Administrator authority that Deployer can use to access the server.

Attribute	Description
pwd	<p>Password associated with the user name. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p>
pwdHandle	<p>The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code>. For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189.</p>
useSSL	<p>Whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication.
version	<p>Version of the server.</p>
install Deployer Resource	<p>Whether Deployer should install the WmDeployerResource package on each Integration Server server that will run the process steps. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to install the package. ■ <i>false</i> to avoid installing the package.
Test	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.
executeACL	<p>Optional. Specifies the ACL for the alias. If no value is specified, Deployer assigns the alias to the Administrators ACL by default.</p>

Setting Up Aliases for Source and Target My webMethods Servers

The following example illustrates how to set up aliases for source and target My webMethods Servers.

```
<MWS>
  <mwsalias name="alias
name ">
    <host>My
webMethods Server host </host>
    <port>My
webMethods Server port </port>
    <user>user
name </user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <excludeCoreTaskEngineDependencies>true/false
</excludeCoreTaskEngineDependencies>
    <cacheTimeOut>time</cacheTimeOut>
    <includeSecurityDependencies>true/false</includeSecurityDependencies>
    <rootFolderAliases>folder, folder, folder...</rootFolderAliases>
    <maximumFolderObjectCount>count</maximumFolderObjectCount>
    <enableAddtionalLogging>true/false</enableAddtionalLogging>
    <maxFolderDepth>number_of_assets</maxFolderDepth>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </mwsalias>
</MWS>
```

For information on the values to supply for the tags below, see ["Connecting to My webMethods Servers" on page 59](#).

Attribute	Description
mwsalias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> .

Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the

Attribute	Description
	passwords in the XML file and run Project Automator to encrypt the passwords again.
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
excludeCoreTaskEngineDependencies	<p>Whether to exclude Task Engine portlets from the dependencies list for task application assets. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to exclude the portlets. ■ <i>false</i> to include the portlets. <p>This attribute corresponds to the Exclude Core Task Engine Dependencies field.</p>
cacheTimeout	Length of time queries should remain in the cache unless the cache capacity is exceeded. This attribute corresponds to the Cache Timeout field.
includeSecurityDependencies	<p>Whether to include the following in the dependencies list for My webMethods Server assets when creating an MWS deployment set:</p> <ul style="list-style-type: none"> ■ Security realms that contain the assets. ■ User/group/role references in the assets' security ACLs. <p>Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to include the assets. ■ <i>false</i> to exclude the assets. <p>This attribute corresponds to the Include security dependencies field.</p>
rootFolderAliases	My webMethods Server aliases to use as root folders when selecting pages to deploy. Separate the folders using commas. This attribute corresponds to the Root folder aliases field.
maximumFolderObjectCount	Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set. This attribute corresponds to the Maximum Folder Object Count field.

Attribute	Description
enableAdditionalLogging	<p>Whether to log debug information about selected assets to source My webMethods Server logs, and assets that Deployer deploys to target My webMethods Server logs. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Enable additional MWS logging field.</p>
maxFolderDepth	<p>Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set. This attribute corresponds to the Maximum Folder Depth field.</p>
useSSL	<p>Whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	<p>Version of the server. This attribute corresponds to the Version field.</p>
Test	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Source and Target Optimize Servers

The following example illustrates how to set up aliases for source and target Optimize servers.

```
<Optimize>
  <optimizealias name="alias name">
    <host>Integration
  Server host</host>
    <port>Integration
```

```

Server port </port>
  <user>user
name </user>
  <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
  <useSSL>true/false</useSSL>
  <version>version_number</version>
  <Test>true/false</Test>
  </optimizealias>
</Optimize>

```

For information on the values to supply for the tags below, see "[Connecting to Optimize Servers](#)" on page 57.

Attribute	Description
optimizealias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see " Using Handles Instead of Passwords " on page 189.
useSSL	<p>Whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>

Attribute	Description
version	Version of the server. This attribute corresponds to the Version field.
Test	Whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Event Servers

The following example illustrates how to set up aliases for target Event Servers.

Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

```
<EventServer>
  <eventserveralias name="event_server_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </eventserveralias>
</EventServer>
```

For information on the values to supply for the tags below, see ["Connecting to Event Servers" on page 62](#).

Attribute	Description
eventserver alias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.

Attribute	Description
<code>user</code>	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
<code>pwd</code>	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p>
<code>pwdHandle</code>	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
<code>useSSL</code>	<p>Whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
<code>version</code>	Version of the server. This attribute corresponds to the Version field.
<code>Test</code>	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target EDA Deployment Endpoints

The following example illustrates how to set up aliases for target EDA deployment endpoints.

```

<EDA>
  <edaserveralias name="EDA_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </edaserveralias>
</EDA>

```

For information on the values to supply for the tags below, see ["Connecting to EDA Deployment Endpoints"](#) on page 69.

Attribute	Description
edaserveralias name	Name to assign to the EDA deployment endpoint. This attribute corresponds to the Name field.
host	Host name or IP address of the Software AG Platform Manager runtime. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189.
useSSL	Whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. This attribute corresponds to the Use SSL field.

Attribute	Description
version	Version of the server. This attribute corresponds to the Version field.
Test	Whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Target Business Rules Integration Servers

The following example illustrates how to set up aliases for source and target Business Rules Integration Servers.

```
<RulesServer>
  <rulesserveralias name="rules_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </rulesserveralias>
</RulesServer>
```

For information on the values to supply for the tags below, see ["Connecting to Business Rules Integration Servers" on page 70](#).

Attribute	Description
rulesserveralias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to

Attribute	Description
	access the server. This attribute corresponds to the User field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p>
pwdHandle	<p>The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code>. For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189.</p>
useSSL	<p>Whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	<p>Version of the server. This attribute corresponds to the Version field.</p>
Test	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Business Rules My webMethods Servers

The following example illustrates how to set up aliases for source and target Business Rules My webMethods Servers.

```
<RulesServerMWS>
  <rulesmwsserveralias name="rules_mws_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd> OR <pwdHandle>$(PasswordHandle) </pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <rootContext>mws_root_context</rootContext>
    <Test>true/false</Test>
  </rulesmwsserveralias>
</RulesServerMWS>
```

For information on the values to supply for the tags below, see "[Connecting to Business Rules My webMethods Servers](#)" on page 71.

Attribute	Description
rulesmwsserveralias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> .

Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.

Attribute	Description
<code>pwdHandle</code>	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
<code>useSSL</code>	Whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. This attribute corresponds to the Use SSL field.
<code>version</code>	Version of the server. This attribute corresponds to the Version field.
<code>rootContext</code>	The root context path for the server (<code>http://:host_name:port_number/root_context</code>). For example, <code>http://localhost:8585/mws</code> .
<code>Test</code>	Whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Universal Messaging Servers

The following example illustrates how to set up aliases for target Universal Messaging servers for basic authentication, SSL authentication, or neither.

Basic Authentication

The following example illustrates how to set up a Universal Messaging server alias that uses basic authentication.

```
<UniversalMessaging>
  <universalmessagingalias name="server_target">
    <realmURL>URL</realmURL>
```

```

    <useBasicAuth>true</useBasicAuth>
    <version>version</version>
    <user>basic authorization user name</user>
    <pwd>basic authorization password</pwd> OR <pwdHandle>handle</pwdHandle>
    <Test>true/false</Test>
  </universalmessagingalias>
</UniversalMessaging>

```

For detailed information on the values to supply for the following attributes, see ["Connect to Universal Messaging Servers" on page 67](#).

Attribute	Description
universalmessagingalias name	Name to assign to the server. This attribute corresponds to the Name field.
realmURL	The URL of the Universal Messaging realm server. This attribute corresponds to the Realm URL field.
useBasicAuth	Set to <i>true</i> to connect to the Universal Messaging server using basic authentication. This attribute corresponds to the Client Authentication > Basic Authentication option.
version	Version of the server. This attribute corresponds to the Version field.
user	Basic authentication user name. This attribute corresponds to the Username field.
pwd	Basic authentication password. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="667 1388 1365 1591" style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see "Using Handles Instead of Passwords" on page 189 .
Test	Whether Deployer should test the connection to the servers. Set to:

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

SSL Authentication

The following example illustrates how to set up a Universal Messaging server alias that uses SSL authentication.

```
<UniversalMessaging>
  <universalmessagingalias name="server_target">
    <realmURL>URL</realmURL>
    <useSSL>true</useSSL>
    <version>version</version>
    <keyStorePath>Deployer keystore path</keyStorePath>
    <keyStorepassword>Deployer keystore password</keyStorepassword>
    <trustStorePath>Deployer truststore path</trustStorePath>
    <trustStorepassword>Deployer truststore password</trustStorepassword>
    <Test>true/false</Test>
  </universalmessagingalias>
</UniversalMessaging>
```

For detailed information on the values to supply for the following attributes, see ["Connect to Universal Messaging Servers" on page 67](#).

Attribute	Description
universalmessagingalias name	Name to assign to the server. This attribute corresponds to the Name field.
realmURL	The URL of the Universal Messaging realm server. This attribute corresponds to the Realm URL field.
useSSL	Set to <i>true</i> to connect to the Universal Messaging server using SSL authentication. This attribute corresponds to the Client Authentication > SSL check box.
version	Version of the Universal Messaging server. This attribute corresponds to the Version field.

Attribute	Description
keyStorePath	Full path to Deployer's keystore file. This attribute corresponds to the DeployerKeystore field.
keyStorepassword	Password that Deployer uses to access its keystore file. This attribute corresponds to the Keystore Password field.
trustStorePath	Full path to Deployer's truststore file. This attribute corresponds to the DeployerTruststore field.
trustStorepassword	Password that Deployer uses to access its truststore file. This attribute corresponds to the Truststore Password field.
Test	<p>Whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

No Authentication

The following example illustrates how to set up a Universal Messaging server alias that does not use client authentication.

```
<UniversalMessaging>
  <universalmessagingalias name="server_target">
    <realmURL>URL</realmURL>
    <version>version</version>
    <Test>true/false</Test>
  </universalmessagingalias>
</UniversalMessaging>
```

For detailed information on the values to supply for the following attributes, see ["Connect to Universal Messaging Servers" on page 67](#).

Attribute	Description
universalmessagingalias name	Name to assign to the server. This attribute corresponds to the Name field.

Attribute	Description
realmURL	The URL of the Universal Messaging realm server. This attribute corresponds to the Realm URL field.
version	Version of the Universal Messaging server. This attribute corresponds to the Version field.
Test	Whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see "Root Tag" on page 191. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Groups

Use the `<TargetGroup>` tag to define the aliases to include in target groups as follows:

```
<TargetGroup description="target group description" isLogicalCluster="true or
false" name="alias name" type="runtime_type" version="version_number">
<alias>server
alias</alias>
  <alias>server
alias</alias>
  <cluster name="cluster name">server, server, server...</cluster>
  <cluster name="cluster name">server, server, server...</cluster>
</TargetGroup>
```

For information on the values to supply for the attributes below, see ["Creating Target Groups" on page 73](#).

Attribute	Description
description	Description of the target group. This attribute corresponds to the Description field.
isLogicalCluster	(Runtime-based deployment only.) Whether Deployer should automatically roll back the deployment on all servers in the group if deployment fails on any of the servers in the target group. This attribute corresponds to the Rollback All on Failure field. Set to:

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>true</i> to roll back all of the servers in the target group if deployment fails on one of the servers in the target group. ■ <i>false</i> to keep Deployer from rolling back all of the servers in the target group if deployment to one of the servers encounters a failure. <p>Note: The <code>isLogicalCluster</code> attribute is valid for runtime-based deployment only. Deployer ignores this attribute for repository-based deployment.</p>
name	Name to use for the target group. This attribute corresponds to the Name field.
type	<p>The runtime type of the target group. Set to:</p> <ul style="list-style-type: none"> ■ <i>Broker</i> ■ <i>IS</i> ■ <i>MWS</i> ■ <i>Optimize</i> ■ <i>ProcessModel</i> ■ <i>RULES</i> ■ <i>EventServer</i> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> <ul style="list-style-type: none"> ■ <i>EDA</i> ■ <i>UniversalMessaging</i>
version	Version of the server. This attribute corresponds to the Version field.
alias	The aliases of the servers to include in the target group. Each alias should have its own alias attribute.
cluster name	A list of Integration Server or process model clusters.

Creating Projects

You create projects in the `<Projects>` tag as follows:

```
<Projects projectPrefix="string">
  <Project overwrite="true" name="project_name"
description=project_description">
  <ProjectProperties>
    <Property name= "projectLocking">true/false</Property>
    <Property name= "concurrentDeployment">true/false</Property>
    <Property name= "ignoreMissingDependencies">true/false</Property>
    <Property name= "isTransactionalDeployment">true/false</Property>
  </ProjectProperties>
  <{DeploymentSet|DeletionSet}>tags</{DeploymentSet|DeletionSet}>
  <{Component|Composite}>tags</{Component|Composite}>
</Project>
</Projects>
```

Note: You can specify attributes in the `<DeletionSet>` tag for repository-based deployment only.

Note: The `<Component>` and `<Composite>` tags define the components and composites for repository-based deployment projects only. They are not used for runtime-based deployment projects.

The `<Projects>` tag can contain the following attribute:

Attribute	Description
<code>projectPrefix</code>	Optional. Specify a prefix for projects created using Project Automator. For example, if you specified <code><Projects projectPrefix="Auto_"></code> , the names of projects created using Project Automator would be prefixed by "Auto_".

The `<Projects>` tag can contain several `<Project>` tags, one for each project you want Project Automator to create. You can specify the following attributes for the `<Project>` tag:

Attribute	Description
<code>overwrite</code>	Determines how Project Automator proceeds if Deployer already contains a project with the same name as one you define. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to overwrite the project. ■ <i>false</i> keep the project, write an error, and continue to the next <code><Project></code> tag.
<code>name</code>	Name of the project.

Attribute	Description
description	Description of the project.

For repository-based projects, the <ProjectProperties> tag contains the individual project properties that you can edit for the project. If no project properties are specified, the project adopts the settings specified for all projects. For more information about setting project settings for all projects, see ["Setting Default Properties for All Projects" on page 79](#).

Attribute	Description
projectLocking	Indicates whether locking is enabled or disabled for the project. When set to <i>true</i> , locking is enabled for the project.
concurrent Deployment	Indicates whether Deployer deploys assets concurrently. When set to <i>true</i> concurrent deployment is enabled.
ignoreMissing Dependencies	Indicates whether Deployer ignores missing dependencies. When set to <i>true</i> , Deployer ignores missing dependencies for the project. If this attribute is set to <i>false</i> and the project contains missing dependencies, deployment fails.
isTransactional Deployment	Indicates whether Deployer automatically creates a checkpoint prior to delivering and activating deployment and deletion sets. When set to <i>true</i> , transactional deployment is enabled. When transactional deployment is enabled and activation fails, Deployer triggers a roll back automatically and restores the target servers to the state of the prior activation.

The following sections describe the tasks you can perform using tags of the parent <Project> tag.

Defining Deployment and Deletion Sets for Runtime-Based Deployment

You define the deployment and deletion sets for a runtime-based project using the <DeploymentSet> tag. Each tag can include these attributes:

```
<DeploymentSet name="set name" description="set description" type="runtime_type"
mode="{Deploy|Delete}" srcAlias="{source server} alias"
defaultDependencyAction="{Add|Fulladd|Exists|Ignore}"></DeploymentSet>
```

The following table describes the attributes in the <DeploymentSet> tag:

Attribute	Description
name	Name of the deployment or deletion set. For more information, see the description of the Name field in "Creating a Deployment Set" on page 96 or "Creating a Deletion Set" on page 128.
description	Optional. Description for the deployment or deletion set. For more information, see the description of the Description field in "Creating a Deployment Set" on page 96 or "Creating a Deletion Set" on page 128.
type	<p>Runtime type of the server that contains the assets to deploy or delete. Set to:</p> <ul style="list-style-type: none"> ■ <i>Broker</i> ■ <i>IS</i> (For Integration Server and Trading Networks servers.) ■ <i>MWS</i> (Deployment sets only.) ■ <i>Optimize</i> (Deployment sets only.) ■ <i>ProcessModel</i> (Deployment sets only.) ■ <i>RULES</i> (Deployment sets only.) ■ <i>EventServer</i> (Deployment sets only.) <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>EDA</i> (Deployment sets only.) <p>For more information, see the description of the Type field in "Creating a Deployment Set" on page 96 or "Creating a Deletion Set" on page 128.</p>
packageRegExp	(Optional.) Text that the package names must contain in order to be listed. For more information, see the description of the Packages field in "Creating a Deployment Set" on page 96 or "Creating a Deletion Set" on page 128.
otherRegExp	(Optional.) The number of assets to display in the list. See the All other assets field in the "Defining a Deployment Set" on page 95 or "Defining a Deletion Set" on page 127 chapters.
mode	Specifies whether Deployer should create a deployment or deletion set for the project. Set to:

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>deploy</i> to define a deployment set. ■ <i>delete</i> to define a deletion set. <p>For example, to define a deployment set, specify <i>deploy</i> for <code>mode</code> as follows:</p> <pre><DeploymentSet name="depSet1" description="depAssets" packageRegExp="" otherRegExp="" type="IS" mode="deploy or delete" srcAlias="ISserver31,ISserver41" defaultDependencyAction="Add" tnTreeNodeCount="1000"></DeploymentSet></pre>
<code>srcAlias</code>	The server alias for the source server (for deployment sets) or target server (for deletion sets). This attribute corresponds with the Name field described for the runtime server type in "Starting Deployer and Connecting to Servers" on page 53 .
<code>defaultDependencyAction</code>	<p>(Optional.) Specifies how Deployer should handle unresolved dependencies. Set to:</p> <ul style="list-style-type: none"> ■ <i>add</i> to add the referenced asset. ■ <i>fulladd</i> to add the entire package that contains the referenced asset. This value is valid for Integration Server assets only. ■ <i>exists</i> to specify that the referenced asset exists on the target server. <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p>Note: If the dependent asset does not exist on the target server, deployment will fail.</p> </div> <ul style="list-style-type: none"> ■ <i>ignore</i> to ignore unresolved dependencies.

The tags for each type of asset vary. See the sample XML file `ProjectAutomatorSample.xml` in the `Integration Server_directory/instances/instance_name/packages/WmDeployer/config` directory for examples for each type of asset. Additional notes are provided below.

Assets	Notes
All	If an asset is located in a folder, the asset tag must include the <code>folder</code> attribute.
Integration Server	<ul style="list-style-type: none"> ■ On <code><Port></code> asset tags, set the <code>protocol</code> attribute to one of the following: <ul style="list-style-type: none"> ■ <i>HTTP</i> ■ <i>FTP</i>

Assets	Notes
	<ul style="list-style-type: none"> ■ <i>HTTPS</i> ■ <i>FTPS</i> ■ <i>Email</i> ■ <i>FilePolling</i> ■ Package asset tags require the attribute <code>package="name"</code>. ■ Scheduled task assets require the attribute <code>filterBy="name"</code>.
Trading Networks	<p>Sets that include Trading Networks assets must also include the <code>tnTreeNodeCount="number"</code> attribute. This attribute specifies the number of Trading Networks assets Deployer should display. For more information, see the description of the Maximum TN Assets to Display field in "Creating a Deployment Set" on page 96 or "Creating a Deletion Set" on page 128.</p>
My webMethods Server	<p><Rule> asset tags require the attribute <code>folder</code> set to:</p> <ul style="list-style-type: none"> ■ <i>Shell Rules</i> ■ <i>Skin Rules</i> ■ <i>Start Page Rules</i> ■ <i>Rendering Rules</i> ■ <i>Login Page Rules</i> ■ <i>Task Rules\Global Task Rules\Schedule Rules</i> ■ <i>Task Rules\Global Task Rules\Trigger Rules</i>

Defining a Deployment Set for Repository-Based Deployment

You define deployment sets for repository-based deployment as follows:

```
<DeploymentSet autoResolve="full|partial|ignore" description="description"
name="deployment_set" srcAlias="repository"> <Composite name="name"
srcAlias="repository_alias" type="type"/> <Component
componentType="component_type" compositeName="name" name="name"
srcAlias="repository_alias" type="runtime_type"/>
```

The <DeploymentSet> tag must include the `autoResolve`, `description`, `name`, and `srcAlias` attributes. The following table describes each of these attributes.

Attribute	Description
autoResolve	<p>Specifies how Deployer should resolve unresolved dependencies in the deployment set. Set to:</p> <ul style="list-style-type: none"> ■ <i>full</i> to resolve the unresolved dependencies for the deployment set by adding the entire referenced composite. This setting corresponds to the Auto resolve by Composite field in the GUI. For more information, see "Resolving Dependencies Automatically" on page 120. ■ <i>partial</i> to resolve the unresolved dependencies for the deployment set by adding only the referenced assets. Adding only the referenced assets, instead of the entire referenced composite, is referred to as <i>partial deployment</i>. You can use partial deployment for only Integration Server, Trading Networks, or Broker (including JNDI) assets. This setting corresponds to the Auto resolve by Asset field in the GUI. For more information, see "Resolving Dependencies Automatically" on page 120. ■ <i>ignore</i> to ignore unresolved dependencies. This corresponds to manually resolving dependencies by setting the Unset/Add/Ignore column to ignore for an asset. For more information, see "Resolving Dependencies Manually" on page 121.
description	This attribute corresponds with the Description field described in "Creating a Deployment Set" on page 96 .
name	Name of the deployment set. This attribute corresponds with the Name field described in "Creating a Deployment Set" on page 96 .
srcAlias	The repository alias for the component. This attribute corresponds with the Name field described in "Connecting to a Repository for Repository-Based Deployment" on page 72 .

The <Composite> tag describes composites that are part of the deployment set. You can set the following attributes for the <Composite> tag.

Attribute	Description
name	Name of composite from the ACDL. You can use an asterisk (*) as a wildcard character. For example, if you set name to "Deploy*", Project Automator adds all composites with a name beginning with "Deploy" to the deployment set.

Attribute	Description
	To use a wildcard character to return composites that contain an asterisk (*) in the name, you must add an escape character (\) before the asterisk in <code>name</code> (example, <code>*</code>). For example, to add all composites with the name <code>"project*"</code> , you would set <code>name</code> to <code>"project**"</code> . To add all composites with the name <code>"*project"</code> , you would set <code>name</code> to <code>"*project*"</code> .
<code>srcAlias</code>	Alias name of the server on which the component is located. This attribute corresponds with the Name field described in "Connecting to a Repository for Repository-Based Deployment" on page 72 .
<code>type</code>	<p>Runtime type of the asset. Set to:</p> <ul style="list-style-type: none"> ■ <i>ApplicationPlatform</i> ■ <i>BPM</i> ■ <i>Broker</i> ■ <i>EDA</i> ■ <i>EventServer</i> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>IS</i> ■ <i>MWS</i> ■ <i>Optimize</i> ■ <i>RULES</i> ■ <i>TN</i> ■ <i>UniversalMessaging</i> <p>You can use an asterisk (*) as a wildcard character. For example, if you set <code>type</code> to <code>"*"</code>, Project Automator adds composites from all runtimes to the deployment set.</p>

The `<Component>` tag describes the assets that are part of the deployment set. You can set the following attributes for the `<Component>` tag.

Attribute	Description
<code>componentType</code>	Type of asset. For example, <code>isdocumenttype</code> .

Attribute	Description
	<p>You can use an asterisk (*) as a wildcard character. For example, if the <code>componentType</code> attribute is set to <code>"**"</code>, Project Automator adds all asset types to the deployment set.</p>
<code>compositeName</code>	<p>Name of composite in which the asset is located. You can use an asterisk (*) as a wildcard character. For example, if the <code>compositeName</code> attribute is set to <code>"is**"</code>, Project Automator adds all assets with a composite name beginning with <code>"is"</code> to the deployment set.</p> <p>To use a wildcard character to return all composites that contain an asterisk (*) in the name, you must add an escape character (\) before the asterisk in <code>compositeName</code> (example, <code>*</code>). For example, to add assets from composites with the name <code>"project**"</code>, you would set <code>compositeName</code> to <code>"project**"</code>. To add all composites with the name <code>"*project"</code>, you would set <code>compositeName</code> to <code>"*project**"</code>.</p>
<code>name</code>	<p>Name of the asset. You can use an asterisk (*) as a wildcard character. For example, if the <code>name</code> attribute is set to <code>"is**"</code>, Project Automator adds all assets with a name beginning with <code>"is"</code> to the deployment set.</p> <p>To use a wildcard character to return all assets that contain an asterisk (*) in the name, you must add an escape character (\) before the asterisk in <code>name</code> (example, <code>*</code>). For example, to add all assets with the name <code>"project**"</code>, you would set <code>name</code> to <code>"project **"</code>. To add all assets with the name <code>"*project"</code>, you would set <code>name</code> to <code>"*project**"</code>.</p>
<code>srcAlias</code>	Alias name of the host server.
<code>type</code>	<p>Runtime type of the asset. Set to:</p> <ul style="list-style-type: none"> ■ <i>BPM</i> ■ <i>Broker</i> ■ <i>EDA</i> ■ <i>EventServer</i> <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>IS</i> ■ <i>MWS</i>

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>Optimize</i> ■ <i>RULES</i> ■ <i>TN</i> ■ <i>UniversalMessaging</i>
	<p>You can use an asterisk (*) as a wildcard character. For example, if you set <code>type</code> to <code>"**"</code>, Project Automator adds assets from all runtimes to the deployment set.</p>

Defining a Deletion Set for Repository-Based Deployment

You define deletion sets for repository-based deployment as follows:

```
<DeletionSet autoResolve="full" description="description" name="deletion_set">
  <Component componentType="component_type" compositeName="composite_name"
name="name" srcAlias="repository_alias" type="runtime_type"/>
</DeletionSet>
```

Note: Application Platform has different assets when deploying from a repository and when defining a deletion set. Deletion set assets represent the entire parent composite. This is because Application Platform cannot delete individual assets. Therefore you must set `name=compositeName` and `type=compositeType`. Thus there are only three types of deletion-time assets: "bundle", "war", and "properties".

The following table describes the attributes in the `<DeletionSet>` tag:

Note: For a description of the attributes for the `<Component>` tag, see ["Defining a Deployment Set for Repository-Based Deployment" on page 224](#).

Attribute	Description
<code>autoResolve</code>	The <code>autoResolve</code> attribute is optional, but if you supply a value, set it to <i>full</i> . If <code>autoResolve</code> is set to <i>full</i> , Deployer automatically finds dependent assets and adds them to the deletion set. If <code>autoResolve</code> is not supplied and Deployer finds dependent assets, Deployer creates the deletion set with unresolved dependencies and deployment will fail. For more information about the <code>autoResolve</code> attribute, see "Defining a Deployment Set for Repository-Based Deployment" on page 224 .
<code>description</code>	Description for the deletion set. This attribute corresponds with the Description field described in "Creating a Deletion Set" on page 128 .

Attribute	Description
name	Name to use for the deletion set. This attribute corresponds with the Name field described in "Creating a Deletion Set" on page 128 .

Building a Project for Runtime-Based Deployment

You define a build for a runtime-based deployment project as follows:

```
<DeploymentBuild name="build name" description="build description" </DeploymentBuild>
```

Note: If the project already has a build with the same name, Deployer overwrites it.

The following table describes the attributes in the <DeploymentBuild> tag:

Attribute	Description
name	Name of the build. This attribute corresponds to the Name field described in "Creating a Build" on page 138 .
description	Description of the build. This attribute corresponds to the Description field described in "Creating a Build" on page 138 .

Mapping a Project

You define a map to create for a project as follows:

```
<DeploymentMap name="map name" description="map description" ></DeploymentMap>
<MapSetMapping mapname="map name" setName="name of deployment or
deletion set">
  <alias>target
server alias</alias>
  <alias>target
server alias</alias>
  <group>target
group</group>
  <group>target
group</group>
</MapSetMapping>
```

Note: If the project already has a map with the same name, Deployer overwrites it.

The following table describes the attributes in the <DeploymentMap> tag:

Attribute	Description
name	Name of the deployment map. This attribute corresponds to the Name field described in "Mapping a Project to Target Servers and Target Groups" on page 142 .

Attribute	Description
description	Description of the deployment map. This attribute corresponds to the Description field described in "Mapping a Project to Target Servers and Target Groups" on page 142.

Mapping a Runtime-Based Project

You define a deployment map for a runtime-based project with the `<DeploymentMap>` and `<MapSetMapping>` tags as follows:

```
<DeploymentMap name="map name " description="map description "></DeploymentMap>
<MapSetMapping mapname="map name " setName="name of deployment or
deletion set ">
  <alias>target_server_alias</alias>
  <group>target_group</group>
  <group>target_group</group>
</MapSetMapping>
```

Note: If the project already has a map with the same name, Deployer overwrites it.

The following table describes the attributes in the `<DeploymentMap>` tag. Each attribute corresponds to a field in the GUI as described in ["Mapping a Project to Target Servers and Target Groups"](#) on page 142.

Attribute	Description
name	Name of the deployment map. This attribute corresponds to the Name field.
description	Description of the deployment map. This attribute corresponds to the Description field.

The `<MapSetMapping>` tag uses the following attributes to define the aliases contained in each deployment map:

Attribute	Description
mapname	Name of the deployment map. This attribute should match the <code>name</code> attribute of the <code><DeploymentMap></code> tag.
setName	Name of the deployment or deletion set. <ul style="list-style-type: none"> For deployment sets, this attribute should match the <code>name</code> attribute of the <code><DeploymentSet></code> tag as described in "Defining Deployment and Deletion Sets for Runtime-Based Deployment" on page 221 or "Defining a Deployment Set for Repository-Based Deployment" on page 224.

Attribute	Description
	<ul style="list-style-type: none"> For deletion sets, this attribute should match the <code>name</code> attribute of the <code><DeletionSet></code> tag as described in "Defining Deployment and Deletion Sets for Runtime-Based Deployment" on page 221 or "Defining a Deletion Set for Repository-Based Deployment" on page 228.
<code>alias</code>	The server alias of the target server. This attribute corresponds to the target server you select when you click Add Target Server as described in "Mapping a Project to Target Servers and Target Groups" on page 142.
<code>group</code>	The alias of the target group. This attribute corresponds to the target group you select when you click Add Target Group as described in "Mapping a Project to Target Servers and Target Groups" on page 142.

Mapping a Repository-Based Project

You define a deployment map for a repository-based project with the `<DeploymentMap>` and `<MapSetMapping>` tags as follows:

```
<DeploymentMap name="map name " description="map description "></DeploymentMap>
<MapSetMapping mapname="map name " setName="name of deployment or
deletion set ">
  <alias type="alias_type ">target_server_alias</alias>
  <alias logicalServer="logical_server " type="alias_type ">target server alias</alias>
</MapSetMapping>
```

Note: If the project already has a map with the same name, Deployer overwrites it.

The following table describes the attributes in the `<DeploymentMap>` tag. Each attribute corresponds to a field in the GUI as described in ["Mapping a Project to Target Servers and Target Groups"](#) on page 142.

Attribute	Description
<code>name</code>	Name of the deployment map. This attribute corresponds to the Name field.
<code>description</code>	Description of the deployment map. This attribute corresponds to the Description field.

The `<MapSetMapping>` tag uses the following attributes to define the aliases contained in each deployment map:

Attribute	Description
mapname	Name of the deployment map. This attribute should match the <code>name</code> attribute of the <code><DeploymentMap></code> tag.
setName	Name of the deployment or deletion set. <ul style="list-style-type: none"> ■ For deployment sets, this attribute should match the <code>name</code> attribute of the <code><DeploymentSet></code> tag as described in "Defining Deployment and Deletion Sets for Runtime-Based Deployment" on page 221 or "Defining a Deployment Set for Repository-Based Deployment" on page 224. ■ For deletion sets, this attribute should match the <code>name</code> attribute of the <code><DeletionSet></code> tag as described in "Defining Deployment and Deletion Sets for Runtime-Based Deployment" on page 221 or "Defining a Deletion Set for Repository-Based Deployment" on page 228.
alias type	The runtime type of the target server alias. This attribute corresponds to the Select Server list described in "Mapping a Project to Target Servers and Target Groups" on page 142.
alias logicalServer	(BPM assets only.) The logical server alias of the target server. This attribute corresponds to the Physical Server field as described in "Connect to BPM Process Model Servers" on page 65.
alias	The server alias of the target server. This attribute corresponds to the target server you select when you click Add Target Server as described in "Mapping a Project to Target Servers and Target Groups" on page 142.
group type	The runtime type of the target group. This attribute corresponds to the Select Server list described in "Mapping a Project to Target Servers and Target Groups" on page 142.
group logicalServer	(BPM assets only.) The logical server to which the target group is mapped. This attribute corresponds to the target server or target group you select after you click Add Target Group as described in "Mapping a Project to Target Servers and Target Groups" on page 142.
group	The group mapping for assets that are not targeted to a logical server. This attribute corresponds to the target group you select

Attribute	Description
	when you click Add Target Group as described in "Mapping a Project to Target Servers and Target Groups" on page 142.

Creating a Deployment Candidate for Runtime-Based Deployment

You define a deployment candidate for a runtime-based project with the `<DeploymentCandidate>` tag as follows:

```
<DeploymentCandidate name="candidate name" description="candidate description"
buildName="build
to use" mapName="map to use"></DeploymentCandidate>
```

Note: If the project already has a deployment candidate with the same name, Deployer overwrites it.

The following table describes the attributes in the `<DeploymentCandidate>` tag when creating a deployment candidate for runtime-based deployment. Each attribute corresponds to a field in the GUI as described in ["Deploying a Project"](#) on page 154.

Attribute	Description
name	Name of the deployment candidate. This attribute corresponds to the Name field.
description	Description of the deployment candidate. This attribute corresponds to the Description field.
buildName	(Runtime-based deployment only.) Name of the project build to deploy. This attribute corresponds to the Project Build field.
mapName	Name of the deployment map that identifies the target servers to which to deploy the assets. This attribute corresponds to the Deployment Map field.

Creating a Deployment Candidate for Repository-Based Deployment

You define a deployment candidate for a repository-based project with the `<DeploymentCandidate>` tag as follows:

```
<DeploymentCandidate name="candidate name" description="candidate description"
mapName="map to
use"></DeploymentCandidate>
```

Note: If the project already has a deployment candidate with the same name, Deployer overwrites it.

The following table describes the attributes in the <DeploymentCandidate> tag when creating a deployment candidate for repository-based deployment. Each attribute corresponds to a field in the GUI as described in ["Deploying a Project" on page 154](#).

Attribute	Description
name	Name of the deployment candidate. This attribute corresponds to the Name field.
description	Description of the deployment candidate. This attribute corresponds to the Description field.
mapName	Name of the deployment map that identifies the target servers to which to deploy the assets. This attribute corresponds to the Deployment Map field.

Running Project Automator

Go to the *Integration Server_directory/instances/instance_name/packages/WmDeployer/bin* directory and run this command:

```
{projectAutomator.bat|projectAutomatorUnix.sh|projectAutomatorMac.sh} full_path_to_XML_file
```

Before executing the specified XML file, Project Automator validates the XML using the schema named *ProjectAutomator.xsd* in the *Integration Server_directory/instances/instance_name/packages/WmDeployer/config* directory.

A Deploying Process Models with E-Forms

■ Deploying Process Models with E-Forms	236
---	-----

Deploying Process Models with E-Forms

Note: These instructions apply to runtime-based deployment only. For repository-based deployment, if you want to deploy ProcessModels with e-forms, you must configure the e-form listener manually in the target environment. For more information about configuring the e-form listener manually, see *Implementing webMethods Content Service Platform for BPM* (for Content Service Platform) or *Implementing E-form Support for BPM* (for My webMethods Server).

If you are creating a ProcessModel deployment set, the e-forms might trigger process steps in the process models. In this case, follow these instructions:

1. When you define the ProcessModel deployment set, JCR or CSP files will appear as dependencies. You must include them in the ProcessModel deployment set unless they already exist on the target server and they already specify the correct paths to the e-form templates and e-form instances folders. For instructions on resolving dependencies, see ["Resolving Dependencies" on page 113](#).

Note: My webMethods Server assets (that is, the e-form templates and e-form instances folders) do not appear as dependencies.

2. When you map the ProcessModel deployment set to the target servers, substitute the password configuration values for the JCR or CSP files. These passwords, which Process Engines use to connect to the My webMethods Server or Content Service Platform that hosts e-forms, must be correct for the target environment. Also change other JCR or CSP file configuration values to be correct for the target environment if necessary.

Note: Configure the CSP in the target server with the same structure and content as the source server.

3. If you are using e-forms with My webMethods Server, perform the following additional tasks:
 - a. Define an MWS deployment set that contains the e-form templates and e-form instances folders from the My webMethods Server that hosts e-forms in the source environment. In the project settings for the project that contains this deployment set, set the **Export Content (Documents)** property to **Yes** (see the **Export Content (Documents)** property as described in ["Creating a Project" on page 85](#)).

Note: This setting is set to **No** by default. Since project settings affect all deployment sets in the project, make sure that other MWS deployment sets you include in the project can share this setting, or do not include other MWS deployment sets in the project.

- b. Before deploying, go to the My webMethods Server that hosts e-forms in the source environment and delete the contents of the e-form instances folder.

B Deploying Optimize Assets

■ Overview	240
■ Disabling Automatic Execution of DDL Statements	240
■ Deploying Optimize Assets in Static DB Schema Mode	241
■ Optimize Deployment Usage Notes	242

Overview

When you use Deployer to deploy Optimize assets, the process often requires modifications to the Analytic Engine DB schema on the target server. The modifications result from the automatic execution of Data Definition Language (DDL) statements on the database. You can configure the Analytic Engine to run in Static DB Schema mode, where the automatic execution of DDL statements is disabled. For more information, see ["Disabling Automatic Execution of DDL Statements" on page 240](#).

When the Analytic Engine runs in Static DB Schema mode, the deployment process for Optimize assets runs through two stages. This ensures that no unauthorized modification of the database schema is allowed.

- **First stage (DDL Recording)** - during this stage, the deployment process carries out the following actions:
 1. Old versions of the Optimize assets are removed from the Analytic Engine cache.
 2. Data Manipulation Language (DML) statements are executed.
 3. DDL statements are collected but not executed.
- After these procedures finish, you manually execute the Analytic Engine SQL scripts.
- **Second stage (Actual Deployment)** - during this stage, the deployment process carries out the following actions:
 1. Determines if the database has been updated with the SQL scripts as described in the first stage.
 2. Creates the deployed Optimize assets.

Note: In practice, these stages are part of a single procedure, as described in ["Deploying Optimize Assets in Static DB Schema Mode" on page 241](#). However, be aware that you must complete both deployment stages for each Optimize deployment set before you start deploying another set. Otherwise you will create orphaned objects in your database.

Disabling Automatic Execution of DDL Statements

You must disable the automatic execution of DDL statements to be able to safely deploy Optimize deployment sets as described in ["Deploying Optimize Assets in Static DB Schema Mode" on page 241](#).

To disable the automatic execution of DDL statements

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

2. On the Define Environments page, click the name of the environment with which you want to work.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Database Settings**.
5. In the **Database Settings for Analytic Engine** area, click **Disable DDL Statements** to disable the automatic execution of DDL statements.
6. Click **Save** to save changes.

Note: If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. Click **Deploy All** to deploy all configuration files to all logical servers in an environment, or click **Deploy Updates** to deploy only the modified configuration files to the affected logical servers in the environment.

The status of the deployment operation appears after the operation is completed. The status includes a list of the files that were deployed to the environment and also lists any errors that occurred.

Note: If you edit a configuration and deploy only the updates, the logical servers in the environment must be restarted for the new configuration settings to take effect.

Deploying Optimize Assets in Static DB Schema Mode

You deploy Optimize assets following the procedure below only when Analytic Engine runs in Static DB Schema mode. For deploying Optimize assets when Static DB Schema mode is not activated, see ["Deploying a Project" on page 151](#).

To deploy Optimize assets when Analytic Engine runs in Static DB Schema mode

1. Make sure you have deployed an environment with Analytic Engine running in Static DB Schema mode as described in ["Disabling Automatic Execution of DDL Statements" on page 240](#).
2. In Integration Server: **Solutions > Deployer**.
3. On the **Deployer > Settings** page, in the **General Deployment Defaults** area, make sure the **Batch Size** property is set to 0 to avoid batching of the deployment set.
4. Go to the **Deployer > Projects** page and configure an Optimize deployment candidate as described in ["Creating a Project" on page 85](#).
5. Click  **Deploy**. Deployer displays the **Projects > project > Deploy** page and lists all deployment candidates that exist for the selected project.

6. To deploy the project, click  in the **Deploy** column for the deployment candidate.

Deployer does the following:

- Removes older versions of the assets in the project build from the target servers to prepare the servers for deployment.
- Creates a deployment report and lists the report in the **Deployment History** area. The deployment report contains the location of the .sql file that you need to execute manually before continuing with the second deployment phase.

7. Locate and execute the SQL scripts.

You must have database administrator privileges on the database to be able to execute the scripts. You will find the script(s) that must be executed in a .sql file on the Analytic Engine host file system. The exact location is specified in the deployment report in the **Deployment History** area.

8. Click  in the **Deploy** column to complete the deployment.

Optimize Deployment Usage Notes

The purpose of this topic is to help you understand various important points concerning the deployment of Optimize assets in Static DB Schema mode.

There are two prerequisites critical to the successful deployment of Optimize assets to a target Analytic Engine running in Static DB Schema mode:

1. For each deployment set you want to deploy, you must always execute the following operations in this order:
 - a. Execute the DDL recording stage.
 - b. Execute all generated SQL statements.
 - c. Execute the actual deployment stage.
2. Ensure that each asset included in the process is removed or deployed, instead of updated.

The Analytic Engine attempts to remove all assets within each deployment set in reverse order and then attempts to deploy them in the specified order. An asset cannot be removed if there are other assets depending on it.

Here is the Optimize assets dependency graph:

```

      / Hierarchy \
ILink - Dimension { } KPI - Rule
      \ EventMap /      /      \      \ DataFilter /
  
```

If the Analytic Engine does not run in Static DB Schema mode and some assets cannot be removed, Deployer attempts to update these assets at deployment time. This is not supported by the Static DB Schema mode.

Potential Problems

When you deploy Optimize assets, it is possible to overlook settings, configure various settings incorrectly, or attempt unsupported actions. As a result, the deployment fails or other issues occur. This topic covers commonly encountered problems.

Deployer Batch Size

Deployer **Batch Size** must always be 0 (zero). If the batch size is set to any other value, Deployer will try to split your deployment set into smaller chunks, which will violate both prerequisites described above.

For more information about setting up the Batch Size property, see ["Setting General Deployment Defaults" on page 79](#).

Removing Assets from a Deployment Set

If you decide that you no longer need an asset that has been previously deployed, you cannot simply remove the asset from your previously deployed deployment set and re-deploy it. If you do so, the existing asset on the target Analytic Engine (which is no longer in the deployment set) will prevent the removal of any assets it depends on.

To address this, you must first remove all assets from the target Analytic Engine that are depended on by the asset you want to remove from your deployment set. To do so:

1. Set up your target host as a source and create a new project with a deployment set containing all the assets you want to remove.
2. Set up your target host as a target again and map the deployment set created in step 1 to it.

Note: For example, you might have two Optimize servers configured with different names and the same host and port.

3. After you create a deployment candidate, click **Create Checkpoint** to activate the **Rollback** button.
4. Click **Rollback** to remove all assets in the deployment set created in step 1 from the target Analytic Engine.

After all depended-on assets are removed, you can deploy the modified deployment set from your original source.

Two or More Deployment Sets for the Same Analytic Engine Using One Deployment Map

This is not supported. You can carry out either of the following supported methods as an alternative:

- Split the Optimize deployment sets into separate projects and complete both stages consecutively for each project.
or
- Remove the target Analytic Engine for all deployment sets. For each deployment set with no target specified in the deployment map, do the following:
 1. Add a target Analytic Engine in the deployment map.
 2. Execute both deployment sets and the generated DDLs in the correct order, as described in Step 1 in "[Optimize Deployment Usage Notes](#)" on page 242.
 3. Remove the target Analytic Engine from the deployment map.

Executing DDL Statements for Two or More Analytic Engines

This situation applies when you have executed the DDL recording stage for one deployment set, and then attempt to execute the DDL recording stage for a second deployment set, with both deployment sets targeting the same Analytic Engine.

This is not supported. You must execute both deployment stages and the generated DDLs for each deployment set in the correct order, as described in Step 1 in "[Optimize Deployment Usage Notes](#)" on page 242.

C Deploying to Clustered Integration Servers

■ Overview	246
■ Setting Up Connections to Integration Servers in the Cluster	246
■ Creating the Target Group	247

Overview

Deployer can deploy assets to clustered Integration Servers and to Trading Networks and process models running on Integration Servers. Keep the following points in mind when deploying to clustered Integration Servers:

- Before you can deploy to a cluster, you must define the connections to the Integration Servers in the cluster as remote servers and identify the Integration Servers as part of a target group in Deployer. You can then map and deploy to the target group as you would to any other target group. Since Trading Networks and process models run on clustered Integration Servers, to deploy Trading Networks assets and process models to a cluster, you must set up connections to the cluster and create a target group that includes the servers in the cluster in the same way.
- Before deploying Trading Networks assets to Trading Networks running on clustered Integration Servers, make sure the `tn.cluster.sync.remoteAliases` property is set for each Trading Networks server in the cluster. For instructions, see *webMethods Trading Networks Administrator's Guide*.
- To deploy process models to Process Engines running on clustered Integration Servers, you must configure each Integration Server hosting a process model in a cluster to use the same alias name and port number as the remote alias defined for the cluster.

Setting Up Connections to Integration Servers in the Cluster

To set up connections to the Integration Servers in the cluster

1. Make sure all Integration Servers in the cluster are up and running.
2. In the Integration Server Administrator for the Integration Server that hosts Deployer, define every Integration Server in the Integration Server cluster as a remote server. For more information about remote servers, and instructions on defining them, see *webMethods Integration Server Administrator's Guide*.

Note: All remote servers should have the same port number as the remote alias of the primary port of the cluster.

3. In Deployer, install the `WmDeployerResource` package on each Integration Server in the Integration Server cluster as follows:
 - a. Go to the **Servers > IS & TN** page; the page lists all Integration Servers you defined as remote servers.
 - b. In the **Install** column, select the check box next to each Integration Server.
 - c. Click **Install**.

Creating the Target Group

In a clustered environment, the only nodes from which you can select servers for the target group are those that are part of the primary configured port. To ensure that the servers are available for the target group, configure all servers in the target group to use the remote alias of the primary port.

Perform the following procedure to create a target group for a clustered environment.

To create a target group in a clustered environment

1. In Deployer, perform one of the following:

For...	Perform the following...
Integration Server or Trading Networks	Go to Target Groups > IS & TN and click Create IS & TN Target Groups .
process models	Go to Create BPM(ProcessModel) Target Group and click Create BPM(ProcessModel) Target Group .

Note: You should not use concurrent deployment when deploying BPM process models to BPM target groups. For more information about concurrent deployment, see "[Concurrent and Sequential Deployment](#)" on page 21.

2. In the **Name** box, type the name to use for the target group. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:
\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
3. In the **Description** box, type a description for the target group. The description length has no limit and can include any characters.
4. In the **Version** box, enter the version of the target group.
5. Click **Create**.
6. In the left-hand pane, click the name of the target group from the **Group Name** column.
7. On the **Configure Target Group** pane, set **Roll Back All on Failure** to **Yes**.

Note: **Roll Back All on Failure** is valid for runtime-based deployment only. Deployer ignores this setting for repository-based deployment.

8. The **Available Servers** list shows the cluster name as a top-level node in the tree, and then all the servers in that cluster as child nodes under the cluster name. Select the

cluster name node, and then click **Add**. The entire cluster tree moves to the **Selected Servers** list. Click **Save**.

Note: If you select individual nodes of the cluster and not the entire cluster, when you deploy, the nodes in the cluster will no longer be identical. Tasks will not run equally well on all servers in the cluster, which could cause errors and failures.

<u>If a server in the cluster is...</u>	<u>Then...</u>
Defined as a remote server alias	The child node for that server shows the remote server alias, and you can select it.
Not defined as a remote server	The child node shows the server host and port, but you cannot select it.
Not running	That server does not appear in the list at all. Ensure that every server in the cluster is defined as a remote server and is up and running. For instructions on defining remote servers, see " Connecting to Integration Servers and Trading Networks Servers " on page 56 and <i>webMethods Integration Server Administrator's Guide</i> .

9. You can add other clusters or individual servers to the target group.
10. Map the project to the target group.
11. Checkpoint, deploy, and, if necessary, roll back the project as you would in an unclustered environment.

D Deployable Assets

■ Application Platform Assets	250
■ BPM Process Development Assets	252
■ Broker Assets	253
■ Business Rules Assets	254
■ EDA Assets	255
■ Event Server Assets	256
■ Integration Server Assets	257
■ Mobile Support Assets	308
■ My webMethods Server Assets	308
■ Optimize Assets	313
■ Trading Networks Assets	314
■ Universal Messaging Assets	317
■ Other Assets	320

Note: The assets listed in this appendix are supported by repository-based deployment.

Application Platform Assets

This section identifies the Application Platform assets that you can deploy using Deployer.

The Application Platform assets types can be:

- Bundle - An OSGi bundle.
- Service - An OSGi service.
- WebApp - A web application.
- Config - An OSGi dynamic configuration because these assets represent an OSGi-specific item unlike the web application.
- JndiResource - A JNDI resource injection. This asset defines how to inject an OSGi dynamic configuration into the JNDI space of a web application. This asset does not represent the configuration itself.

Application Platform composites may contain the following assets:

- bundle composite: Bundle, WebApp, Service
- war composite: WebApp
- properties composite: Config, JndiResource

Note: You can deploy Application Platform assets in repository-based deployment only.

The following table lists:

- The assets that you can export
- The asset type ID for each asset
- Asset dependencies

Asset	Asset Type ID	Dependencies, Substitutions, and Other Considerations
OSGi Bundle	Bundle	Each "bundle" composite always contains exactly one Bundle asset. The Bundle asset can depend on zero or more Service assets that are almost always provided by other "bundle" composites.

Asset	Asset Type ID	Dependencies, Substitutions, and Other Considerations
OSGi Service	Service	Each "bundle" composite can contain zero or more Service assets. The Service asset does not have any dependencies on other assets.
OSGi Dynamic Configuration	Config	Each "properties" composite always contains exactly one Config asset. The Config asset does not have any dependencies on other assets. By default, all properties in the "properties" file represented by the "properties" composite are translated to substitutions for the Config asset.
JNDI resource injection	JndiResource	<p>When a "properties" composite defines a JNDI resource that needs to be injected into a web application, it will contain exactly one JndiResource asset in addition to the main Config asset.</p> <p>When the JndiResource asset defines an injection of an OSGi Service into the web application, it will have a dependency on the respective Service asset. JndiResource assets that do not define a Service injection can never have dependencies on other assets.</p> <p>When a "properties" composite contains a JndiResource, all properties of the respective properties file except those that describe the JNDI name and target web application context, are translated to substitutions for the Config asset of the "properties" composite.</p>
Web Application	WebApp	The "war" composite always contains exactly one WebApp asset. The "bundle" composite can contain a single WebApp asset if the composite represents a "web bundle" (WAB). The WebApp asset can have dependencies on one or more JndiResource assets - one for every JNDI resource described in the web.xml.

BPM Process Development Assets

You can prepare BPM Process Development process models (.process files) created with Software AG Designer for deployment with Deployer. The build process enables you to filter the deployable assets by process ID, process version, or a combination of the two.

The following table lists:

- The asset that you can export
- The asset type ID
- Asset dependencies

Asset	Asset Type ID	Description
Process model	bpmprocess	<p>A typical business process model contains references to and dependencies on a number of other runtime assets. For successful deployment, each of these other assets must be checked in to your version control repository. Before you attempt to deploy a business process model, ensure that you have all process elements and dependent assets checked into your repository, including:</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note:Deployer cannot determine dependencies for dynamically referenced processes. You should manually deploy such dependencies.</p> </div> <ul style="list-style-type: none"> ■ The entire process project, including the project directory, the build.xml file, and all .process files. ■ The generated process package in Integration Server. ■ Any Integration Server packages that contain Integration Server assets (such as IS documents or services, including Blaze rule services) that are invoked from the process. ■ Any Composite Application Framework (CAF) assets (especially tasks), any rule assets, Trading Networks assets, referenced process assets from other process projects, or other assets that are invoked from the process.

Broker Assets

Using Deployer, you can deploy your Broker assets and JNDI assets to another Broker, and share the assets that you created in Broker with other applications.

webMethods Broker enables you to export the following assets to Deployer:

- Broker assets such as clients, client groups, and document types.
- JNDI assets such as JMS queues and JMS topics created by webMethods JNDI providers.

Some Broker assets have dependencies on other Broker components. When you export assets that have dependencies on other assets, the corresponding dependent assets are also exported. For example, when you export a client group, the document types belonging to that client group are also exported.

The following table lists:

- The assets that you can export
- The asset type ID for each asset
- Asset dependencies

Asset	Asset Type ID	Dependencies
Client group	ClientGroup	Document Type
Client	Client	Client Group
Document type	DocumentType	None
JMS destination (JMS queues and JMS topics created by webMethods JNDI providers)	JMSDestination	None

Business Rules Assets

The following table describes the user-created assets that you can include when using webMethods Business Rules in deployment projects.

Note: You can deploy business rules assets in repository-based deployment only.

The following table lists:

- The assets that you can export
- The asset type ID for each asset
- Asset dependencies

Asset	Asset Type ID	Dependencies								
Action	ruleaction	<p>Actions have dependencies based on the type of action as follows:</p> <table border="1"> <thead> <tr> <th>For this type of action...</th> <th>The dependency is...</th> </tr> </thead> <tbody> <tr> <td>Service</td> <td>Integration Server service</td> </tr> <tr> <td>Process</td> <td>BPM process</td> </tr> <tr> <td>New Data</td> <td>Business rules data model</td> </tr> </tbody> </table>	For this type of action...	The dependency is...	Service	Integration Server service	Process	BPM process	New Data	Business rules data model
For this type of action...	The dependency is...									
Service	Integration Server service									
Process	BPM process									
New Data	Business rules data model									
Data model	ruledatamodel	Optionally dependent on a nested data model.								
Decision table	ruledecisiontable	<p><i>Always</i> depends on at least one data model.</p> <p>Optionally dependent on one or more action.</p>								
Event rule	ruleeventrule	<p><i>Always</i> dependent on at least one data model.</p> <p>Optionally dependent on one or more action.</p>								

Asset	Asset Type ID	Dependencies
Rule sets	ruleset	<p><i>Always</i> dependent on one of the following:</p> <ul style="list-style-type: none"> ■ Decision table ■ Event rule
Rule project	ruleproject	<ul style="list-style-type: none"> ■ com.softwareag.rules.hotdeploy.project When you deploy on My webMethods Server, hot deploy the rule project to pre-configured Integration Server connection(s). Valid values are <code>true</code> and <code>false</code>. By default, the value is <code>false</code>. ■ com.softwareag.rules.merge.project When you deploy on My webMethods Server, merge the rule project with the existing rule project that has the same name in the repository. Valid values are <code>true</code> and <code>false</code>. By default, the value is <code>false</code> <p>For more information, see <i>Working with Business Rules in My webMethods</i>.</p>

EDA Assets

This section identifies the assets that you can deploy to EDA deployment endpoints using Deployer.

Note: You can deploy EDA assets in repository-based deployment only.

The following table lists:

- The asset that you can export
- The asset type ID
- Asset dependencies

Asset	Asset Type ID	Dependencies
Event Type schema definitions	XSD	None.

Event Server Assets

This section identifies the assets that you can deploy to Event Server using Deployer.

Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

You can deploy the following Event Server assets:

- Continuous query projects
- Event type projects

Note: You can deploy Event Server assets in repository-based deployment only. You cannot deploy Event Server assets in runtime-based deployment.

The following table lists the dependencies and properties that you can substitute when using Deployer to deploy assets to an Event Server.

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations				
Continuous query projects	XML	<p>Dependencies:</p> <p>None.</p> <p>Substitution values:</p> <p>For each database source defined in the project, you can substitute the following properties:</p> <table border="1"> <thead> <tr> <th>Property</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>JdbcConnectionUrl</td> <td>The URL to be used to connect to the database.</td> </tr> </tbody> </table>	Property	Description	JdbcConnectionUrl	The URL to be used to connect to the database.
Property	Description					
JdbcConnectionUrl	The URL to be used to connect to the database.					

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>Username</p> <p>The user name to be used to connect to the database.</p>
		<p>Password</p> <p>The password for the user specified in Username.</p>
		<p>A database source in the composite file has the same URL and user name values that it did at design time. These values become the default values for the JdbcConnectionUrl and Username properties at deployment time. If you leave the JdbcConnectionUrl and Username property fields empty at deployment time, the default values (those that were assigned to the database source at design-time) are assigned to the database source when it is deployed. The password for a database source, however, is never included in the composite file. Therefore, the Password property does not have an associated default value that can be used for deployment. Because it has no default value, you must explicitly assign a value to the Password property for each database source in the composite file.</p> <p>Note: If multiple database sources use the same Password value, select all of them in the Configurable Components panel, specify the password value and then click Save Substitutions. Doing this will assign the specified password to all of the selected data sources in one step.</p>

Integration Server Assets

This section identifies the assets that you can export from Integration Server and deploy using Deployer.

The assets Integration Server supports for exporting belong to the following asset group types.

- Integration Server administrative assets such as ACLs, extended settings, groups, JMS alias, JNDI aliases, ports, scheduled tasks and users.

Note: Integration Server does not generate any assets for the Native Users, ACLs, or Groups. This includes the following:

- **Native Users.** Administrator, Default, Replicator, and Developer.
 - **Native Groups.** Everybody, Administrators, Anonymous, Developers, and Replicators.
 - **Native ACLs.** Administrators, Anonymous, Replicators, Developers, Default, and Internal.
- Integration Server packages.
 - Adapter runtime assets, used by all WmART based adapters and .NET asset, used by webMethods Package for Microsoft .NET.

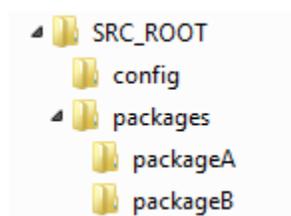
Integration Server Administrative Assets

This section describes the following:

- [Adding Administrative Assets to the Source Directory](#)
- [Global Values for Integration Server Administrative Assets](#)
- [Integration Server Administrative Assets and Substitution Values](#)

Adding Administrative Assets to the Source Directory

To include administrative assets in the composite for deployment, you must manually copy or check in the *Integration Server_directory\instances\instance_name\config* folder to the source directory. Software AG recommends that you structure your source directory to contain all of the administrative assets to be included in the composite as shown in the following example:



In this example, you would define SRC_ROOT as the value of the build.source.dir property in the build.properties file. For more information about the build.properties file, see ["Setting Build Properties" on page 32](#).

When run, the build script creates the following:

- A composite named *package_name.zip* for each package included in the source directory, where *package_name* is the name of the package (with a composite type ID of ispackage). For this example, the files would be named packageA.zip and packageB.zip.

- A composite named `isconfiguration.zip` (with a composite type ID of `isconfiguration`) that contains the administrative assets contained in the `config` directory.

For more information about building composites for repository-based deployment, see ["Building Composites for Repository-Based Deployment" on page 29](#).

The following table lists the files and directories that you must manually copy or check in to the source directory in order to build Integration Server administrative assets for deployment.

Asset	Files to copy or check into the source directory
ACLs	<ul style="list-style-type: none"> ■ <code>Integration Server_directory\instances\instance_name \config \users.cnf</code> ■ <code>Integration Server_directory\instances\instance_name \config \acls.cnf</code>
Broker settings	<code>Integration Server_directory\instances\instance_name \config \dispatch.cnf</code>
Cache manager	<p>All Ehcache configuration files located in the <code>Integration Server_directory\instances\instance_name \config \Caching</code> directory.</p> <p>Note: Do not include system cache managers. For example, do not include cache managers whose names start with "SoftwareAG".</p>
Certificate settings	<code>Integration Server_directory\instances\instance_name \config \server.cnf</code>
Client certificate	<p>The Asset Build Environment extracts this asset from the database.</p> <ul style="list-style-type: none"> ■ If you are using the embedded database, the Asset Build Environment must have access to <code>Integration Server_directory\instances\instance_name \db \embedded</code> directory. ■ If you are using an external database, the Asset Build Environment requires access to the JDBC configuration files.
CSRF guard configuration	<p><code>Integration Server_directory\instances\instance_name \config \security\csrf\csrfguard.cnf</code></p> <p>Note: Integration Server creates this file only when the CSRF guard option is enabled in Integration Server</p>

Asset	Files to copy or check into the source directory
	Administrator. For more information, see <i>webMethods Integration Server Administrator's Guide</i> .
Enhanced parser	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \parsing.cnf
Enterprise Gateway configuration	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \security\enterprisegateway\enterpriseGatewayRules.cnf
Extended settings	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \server.cnf
Global variables	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \globalVariables.cnf
Groups	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \users.cnf
IS Packages	<p>For all the assets contained in the IS package ACDL file, you must check in the <i>Integration Server_directory</i>\instances\<i>instance_name</i> \packages\<i>package_name</i> directory (where <i>package_name</i> is the package for which the ACDL is required) to the source directory.</p> <p>The following files are required to retain ACL information for the different assets in a package:</p> <ul style="list-style-type: none"> ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \config \acls.cnf ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \config \aclmap_sm.cnf ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \config \acllist.cnf ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \config \aclread.cnf ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \config \aclwrite.cnf
JDBC driver alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \jdbc \driver*.xml

Asset	Files to copy or check into the source directory
	<p>Note: The Asset Build Environment does not extract default driver aliases that are installed with Integration Server.</p>
JDBC functional alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\jdbc\function*.xml
JDBC pool alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\jdbc\pool*.xml
	<p>Note: The Asset Build Environment does not extract the embedded database pool alias.</p>
JMS aliases	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\jms.cnf
JNDI aliases	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\jndi\jndi_*.properties
Keystore alias	The Asset Build Environment extracts keystore alias details from the *_config.xml files stored in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\security\keystore directory and then reads the actual keystore binary file (the .jks or .p12) from the value found in the "location" field of the keystore alias definition.
LDAP configuration	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\ldap.cnf
Metadata	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmAssetPublisher\config\assetpublisher.cnf
Mobile Support configuration	<ul style="list-style-type: none"> ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \packages\WmMobileSupport\config\mobileSyncComponents.cnf ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \packages\WmMobileSupport\config\mobileApp.cnf <p>For more information about these assets, see "Mobile Support Assets" on page 308.</p>
Ports	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\ <i>package_name</i> \config\listeners.cnf

Asset	Files to copy or check into the source directory
Proxy server alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \proxy.cnf
Proxy server bypass	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \server.cnf Note: The Asset Build Environment does not extract this file if <code>watt.net.proxySkipList</code> is set to <code>localhost</code> . For more information about <code>watt.net.proxySkipList</code> , see <i>webMethods Integration Server Administrator's Guide</i> .
Quiesce mode configuration	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \quiesce.cnf
Reliable messaging configuration	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \reliableMessaging.cnf
Remote server alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \remote.cnf Note: The Asset Build Environment does not extract remote server aliases named "local".
SAML token issuer	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \security\saml\trusted_saml_issuers.cnf
Scheduled tasks	The Asset Build Environment extracts this asset from a database, and requires either of the following: <ul style="list-style-type: none"> ■ If you are using the embedded database, the Asset Build Environment must have access to <i>Integration Server_directory</i>\instances\<i>instance_name</i> \db \embedded directory. ■ If you are using an external database, the Asset Build Environment requires access to the JDBC configuration files.
SFTP server alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\sftp \sftpServerAliases.cnf
SFTP user alias	■ <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \sftp\sftpUserAliases.cnf

Asset	Files to copy or check into the source directory
	<ul style="list-style-type: none"> ■ <i>Integration Server_directory</i>\instances\<i>instance_name</i> \config \sftp\identities directory and its contents
Truststore alias	The Asset Build Environment extracts truststore alias details from the *_config.xml files stored in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \security\keystore directory and reads the actual truststore binary file (.jks or .p12) from the value found in the "location" field of the truststore alias definition.
URL alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages \package_name \config\urlalias.cnf
Users	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \users.cnf
Integration Cloud Accounts	<i>Integration Server_directory</i> \config\integrationlive \connections.cnf
Integration Cloud Applications	<i>Integration Server_directory</i> \config\integrationlive\applications directory
Integration Cloud Settings	<i>Integration Server_directory</i> \config\integrationlive\accounts.cnf
Web service endpoint alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config \endpoints*.cnf
Web service policy	All .policy files in the <i>Integration Server_directory</i> \instances \ <i>instance_name</i> \config\wss\policies directory.

Global Values for Integration Server Administrative Assets

The following table lists the global administrative assets and substitution values for each that Integration Server supports for deployment. The assets presented in the table are deployed as assets within the isconfiguration composite.

Asset	Asset Type ID	Substitution Values
Cache manager	iscachemanager	<p>Reload Cache Managers After Deployment (<code>reloadCacheManagersAfterDeployment</code>)</p> <p>Specifies whether the cache managers on the target servers are started after deployment.</p> <ul style="list-style-type: none"> ■ True starts the cache manager on the target servers after deployment. If the cache manager is already in the start state on the target Integration Server, a value of True will restart the cache manager with the new configuration. ■ False does not start the cache manager on the target servers after deployment. If the cache manager is already in the start state on the target Integration Server, a value of False will not restart the cache manager. ■ None uses the value specified at the cache manager level Reload property. If the Reload property of the cache manager has the value None, cache manager is not started after deployment. <p>If the cache manager is in a start state on the target Integration Server, a value of None will not restart the cache manager. If the cache manager is in a shutdown state on target Integration Server, the cache manager will not be started.</p> <p>The default is None.</p>
<p>Note: Integration Server deploys a cache manager that uses BigMemory or Terracotta Server Array only if you have the appropriate Terracotta and Integration Server licenses. For more information about licenses, see <i>webMethods Integration Server Administrator's Guide</i>.</p>		
Ports	isport	<p>Enable Ports After Deployment (<code>enablePortsAfterDeploy</code>)</p> <p>Specifies whether the ports on the target servers are enabled after deployment.</p>

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ True enables the port on the target servers after deployment. ■ False disables the port on the target servers after deployment. ■ None uses the value specified at the port level Enable property. The default is None.
Scheduled task	istask	<p data-bbox="781 615 1182 678">Suspend tasks during deployment (suspendTasksDuringDeploy)</p> <p data-bbox="781 699 1328 762">Specifies whether you want to suspend all scheduled tasks during deployment.</p> <ul style="list-style-type: none"> ■ True indicates that you want to suspend all tasks during deployment. ■ False indicates that you want all tasks to run as scheduled during deployment. ■ None indicates that you want use the value specified by the Suspend During Deployment value of each scheduled task asset. If the task has the value of None, no action is taken during deployment. The default value is None. <hr/> <p data-bbox="781 1213 1149 1276">Activate tasks after deployment (activateTasksAfterDeploy)</p> <p data-bbox="781 1297 1317 1360">Specifies whether you want to activate all scheduled tasks after deployment.</p> <ul style="list-style-type: none"> ■ True indicates that you want all tasks to activate after deployment. ■ False indicates that you want all tasks to suspend after deployment. ■ None indicates that you want use the value specified by the Activate After Deployment value from each scheduled task asset. If the task has the value of None, no action will be taken during deployment.

Integration Server Administrative Assets and Substitution Values

The following sections describe the Integration Server administrative assets and their corresponding substitution values.

ACLs

The following table lists the asset type ID and substitution values for ACL assets.

Asset Type ID	isacl
Substitution Values	
None.	

Broker Settings

The following table lists the asset type ID and substitution values for Broker settings assets.

Asset Type ID	isbrokersettings
Substitution Values	
Broker Configuration Settings:	
Broker Host brokerHost	Name (<i>DNSname:port</i> or <i>ipaddress:port</i>) of the machine on which the Broker Server resides.
Broker Name brokerName	Name of the Broker to which the Integration Server connects.
Client Group clientGroup	Client group to which the Integration Server belongs.
Client Prefix CLIENTPREFIX	A string that identifies the Integration Server to the Broker.
Client Authentication Settings:	
Username	User name required to connect to the Broker.

<code>brokerUser</code>	
Password	Password required to connect to the Broker.
<code>brokerPassword</code>	
Keystore	The full path to the keystore file for the target Integration Server.
<code>certFile</code>	
Keystore Password	Password required to access the SSL certificate in the Integration Server's keystore file.
<code>password</code>	
Use Source Keystore	Specifies whether the target Integration Server uses the keystore file from the source Integration Server.
<code>useSource BrokerKeystore</code>	<ul style="list-style-type: none"> ■ True indicates that the target Integration Server uses the keystore file from the source Integration Server. This is the default. ■ False indicates that the target Integration Server will use a different keystore file than the source Integration Server.
	Note: This configuration value does not correspond to a field or property in Integration Server.
Encryption Settings:	
Truststore	Full path to the Integration Server's client truststore file.
<code>truststore</code>	
Use Source Truststore	Specifies whether the target Integration Server uses the truststore file from the source Integration Server.
<code>useSource Broker Truststore</code>	<ul style="list-style-type: none"> ■ True indicates that the target Integration Server uses the truststore file from the source Integration Server. This is the default. ■ False indicates that the target Integration Server will use a different truststore file than the source Integration Server.
	Note: This configuration value does not correspond to a field or property in Integration Server.

Cache Manager

Note: Integration Server deploys a cache manager that uses BigMemory or Terracotta Server Array only if you have the appropriate Terracotta and Integration Server licenses. For more information about licenses, see *webMethods Integration Server Administrator's Guide*.

Asset Type ID `iscachemanager`

Substitution Values

`cacheManagerName`

Name. The name of the cache manager.

Note: If a cache manager with the same name exists in both the source and target servers and if the cache manager name is modified while deploying to the target server, the target server will contain both the cache managers, one with the old name as well as the one with the name used while deploying.

`urls`

Terracotta Server Array URLs. (Optional.) Comma-separated list of host names and port numbers, one for each server in the Terracotta Server Array. You can add multiple URLs by using this format:
`host1:port,host2:port...`

Note: You must specify the Terracotta Server Array URLs only if you have distributed caches in the cache manager.

`reload`

Reload. Specifies whether the cache manager on the target servers is started after deployment.

- **True** starts the cache manager on the target servers after deployment.

- **False** does not start the cache manager on the target servers after deployment.
- **None** indicates that you want to use the global value specified by the deployment options on the **Deployment Map Properties > Configure Builds by Assets** screen and by the **Reload Cache Managers After Deployment** value in the project map file.

If the cache manager is in a start state on the target Integration Server, a value of **None** will not restart the cache manager. If the cache manager is in a shutdown state on target Integration Server, the cache manager will not be started.

The default is **None**.

Certificate Settings

Asset Type ID iscertificates

Substitution Values

None.

Client Certificates

Asset Type ID isclientcerts

Substitution Values

Certificate Path	The name of the file containing the certificate that you want to import. You may specify the file name using an absolute path or using a path that is relative to <i>Integration Server_directory</i> . The file must contain an X.509 certificate in DER file format.
certificate Path	

CSRF Guard Configuration

Asset Type ID	<code>iscsrfguardconfig</code>
Substitution Values	
Enabled	Specifies whether CSRF guard is enabled in Integration Server.
<code>isEnabled</code>	<ul style="list-style-type: none"> ■ True specifies that CSRF guard is enabled. ■ False specifies that CSRF is not enabled. This is the default.
Excluded User Agents	A list of user agents for which Integration Server is not to apply CSRF guard. If CSRF guard is enabled, Integration Server requires that HTTP requests coming from user agents that are not specified as excluded must contain CSRF secure tokens.
<code>excludedUserAgents</code>	
Landing Pages	A list of landing pages for the packages in your Integration Server. A landing page is the home page for a package. Integration Server does not check for CSRF secure tokens in the landing pages, but inserts a token for that page. Integration Server guards all further requests from these landing pages with CSRF secure tokens.
<code>landingPages</code>	
Unprotected URLs	The URLs for which Integration Server is not to check for CSRF secure tokens. If CSRF guard is enabled, Integration Server requires that the requests coming from all URLs that are not specified as unprotected must contain CSRF secure tokens.
<code>unprotectedURLs</code>	
Denial Action	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.
<code>denialAction</code>	<ul style="list-style-type: none"> ■ Error specifies that you want Integration Server to throw an access denied error and terminate the request. This is the default. ■ Redirect specifies that Integration Server is to redirect the user to a confirmation page or the home page of Integration Server Administrator.

webMethods Enterprise Gateway Configuration

Asset Type ID	<code>isenterprisegatewayrules</code>
----------------------	---------------------------------------

Substitution Values

Email To <code>emailTo</code>	One or more email addresses to which the Integration Server acting as the Enterprise Gateway Server sends an alert when a request violates an Enterprise Gateway rule. This is the global value associated with the default alert option. The server uses this setting when a rule does not specify a custom alert setting.
---	---

User Name <code>userName</code>	Integration Server user ID to run the flow service that executes when a request violates an Enterprise Gateway rule. This is the global value associated with the default alert option. Enterprise Gateway Server uses this setting when a rule does not specify a custom alert setting.
---	--

Email To <code>rule_name_emailTo</code>	One or more email addresses to which Enterprise Gateway Server sends an alert when a request violates an Enterprise Gateway rule. This value is part of the custom alert options defined for an individual rule.
---	--

User Name <code>rule_name_userName</code>	Integration Server user ID to run the flow service that executes when a request violates an Enterprise Gateway rule. This value is part of the custom alert options defined for an individual rule.
---	---

Extended Settings

Asset Type ID	<code>isproperty</code>
----------------------	-------------------------

Substitution Values

Server configuration parameters that are set as visible. Allows users to specify values for watt properties that are set as visible in the source Integration Server.

File Access Control Configuration

Asset Type ID	<code>isfileaccesscontrol</code>
----------------------	----------------------------------

Substitution Values

Allowed Read Paths	Semicolon-delimited list of directories to which the services in the <code>pub.file</code> folder in the <code>WmPublic</code> package have read permission.
---------------------------	--

allowedRead
Paths

Allowed Write Paths Semicolon-delimited list of directories to which the services in the pub.file folder in the WmPublic package have write permission.

allowedWrite
Paths

Allowed Delete Paths Semicolon-delimited list of directories that the services in the pub.file folder in the WmPublic package have permission to delete.

allowedDelete
Paths

Note: After making changes and deploying fileAccessControl.cnf, you must reload the WmPublic package or restart Integration Server for the changes to take effect.

Global Variables

Asset Type ID isglobalvariable

Substitution Values

Value Values for global variables.

Value

Groups

Asset Type ID isgroup

Substitution Values

None.

JDBC Driver Alias

Asset Type ID isjdbcdriveralias

Substitution Values

None.

JDBC Pool Alias Configuration

Asset Type `isjdbcpoolalias`
ID

Substitution Values

Database URL The URL to use to connect to the database.
`databaseURL`

User ID Database user for the target Integration Server to use to
communicate with the database.
`userID`

Password Password for the specified database user.
`password`

JDBC Functional Alias

Asset Type `isjdbcfunctionalalias`
ID

Substitution Values

None.

JMS Connection Alias

Asset Type `isjmsalias`
ID

Substitution Values

General Settings:

Connection Client ID The JMS client identifier associated with the connections
established by this JMS connection alias.
`clientID`

User user	User name required to acquire a connection from the connection factory.
---------------------	---

Password password	Password required to acquire a connection from the connection factory.
-----------------------------	--

JNDI Connection Protocol Settings:

JNDI Provider Alias Name jndi_jndiAlias Name	Alias name of the JNDI provider.
---	----------------------------------

JNDI Connection Factory Lookup Name jndi_ connection FactoryLookup Name	Lookup name that the connection factory uses to create a connection to the JNDI provider.
--	---

Native webMethods Connection Protocol Settings:

Broker Host nwm_brokerHost	Name (<i>DNSname:port</i> or <i>ipaddress:port</i>) of the machine on which the Broker Server resides.
--------------------------------------	--

Broker Name nwm_brokerName	Name of the Broker as defined on the Broker Server.
--------------------------------------	---

Client Group nwm_client Group	Name of the client group that you want the target Integration Server to use when it acts as a JMS client.
--	---

Broker List nwm_brokerList	Comma delimited list of Broker Servers on which the connection between the target Integration Server (acting as the JMS client) and the webMethods Broker (acting as a JMS provider) can exist.
--------------------------------------	---

Keystore nwm_keystore	The full path to the target Integration Server's keystore file.
---------------------------------	---

Truststore Full path to target Integration Server client's truststore file.

nwm_truststore

JNDI Alias

Asset Type ID isjndialias

Substitution Values

Provider URL 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.

providerURL

Provider URL Failover List A list of URLs to which the target Integration Server can connect if the connection to the primary JNDI provider becomes unavailable. Separate the URLs with an ampersand, new line, carriage return, or horizontal tab.

providerURL
FailoverList

Security Principal The principal name, or user name supplied by the target Integration Server to the JNDI provider, if the provider requires one for accessing the JNDI directory. For information about whether or not the JNDI provider requires security principal information, consult the product documentation for the JNDI provider.

security
Principal

Security Credentials The credentials, or password, that the target Integration Server provides to the JNDI provider, if the provider requires security credentials to access the JNDI directory. For information about whether or not the JNDI provider requires security credentials, consult the product documentation for the JNDI provider.

security
Credentials

Kerberos Settings

Asset Type ID iskerberosconfig

Substitution Values

Realm The domain name of the Kerberos server, in all uppercase letters. All the computers managed by the

realm

		Key Distribution Center (KDC) and secondary KDCs, if any, constitute the realm.
Key Distribution Center Host		The host name of the machine on which the KDC resides.
<code>kdc</code>		
Kerberos Configuration File		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 hostnames and Kerberos realms mappings.
<code>conf</code>		
Use Subject Credentials		Whether Integration Server requires a Kerberos V5 Generic Security Services (GSS) mechanism to obtain the necessary credentials from an existing subject set up by the JAAS authentication module. Subject represents the user or service being authenticated in the JAAS login context.
<code>useSubjectCredsOnly</code>		
Keystore Alias		
Asset Type	<code>iskeystorealias</code>	
ID		
Substitution Values		
Location		The full path to the keystore file for the target Integration Server.
<code>ksLocation</code>		
Password		Password associated with this alias that is used to protect the contents of the keystore.
<code>ksPassword</code>		

Key Alias Password Password for each key alias residing in the keystore.

keyAlias.
keyAliasName

Use Source Keystore Specifies whether the target Integration Server uses the keystore file from the source Integration Server.

useSource
Keystore

- **True** indicates that the target Integration Server uses the keystore file from the source Integration Server. This is the default.
- **False** indicates that the target Integration Server will use a different keystore file than the source Integration Server.

Note: This configuration value does not correspond to a field or property in Integration Server.

LDAP Configuration

Asset Type ID isldapdirectory

Substitution Values

Directory URL The complete URL of the LDAP server. The URL has the format *protocol:\hostname:portnumber\DistinguishedName*, where:

directoryURL

- *protocol* is ldap for standard connections or ldaps for secure connections

host is the host name or IP address of the LDAP server

portnumber 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.

DistinguishedName is optional, and is in the form of an LDAP distinguished name (DN), for example "dc=webMethods, dc=com", or "o=webMethods.com", depending on how your directory is set up.

The default value for directoryURL is **ldap://localhost:389**.

Principal The user ID the Integration Server should supply to connect to the LDAP server, for example, o=webm.com or dc=webm, dc=com.

principal

Credentials The password the Integration Server should supply to connect to the LDAP server, that is, the Principal's password.

credentials

Metadata

Note: Deployer extracts the metadata asset from the WmAssetPublisher package. Make sure this package resides in the source folder before extracting the asset.

Asset Type ismetadata
ID

Substitution Values

CentraSite URL The CentraSite URL to which to publish metadata about Integration Server assets.

centraSiteURL

For example: `http:\localhost:port\CentraSite\CentraSite`

User Name The name of the user account on CentraSite that will be used for publishing and retracting metadata.

centraSiteUser
name

Password Password associated with **User Name**.

centraSite
Password

Enhanced Parser

Asset Type isparsing
ID

Substitution Values

Default Partition Bytes Specifies the size, measured in bytes, of the partitions on the heap where the enhanced XML parser stores parsed document information. Specify a suffix of "k" to indicate kilobytes or "m" to indicate megabytes. For example, 10k or 10m.

default
PartitionBytes

Note: This value is used only when the enhanced parser is invoked and no partition size is specified.

Use Cache Indicates if caching used with the enhanced XML parser.

<code>useCache</code>	<ul style="list-style-type: none"> ■ True indicates that during parsing of an XML document, Integration Server moves partitions to an off-heap area on disk when the on-heap partition space becomes full and retrieves the partitions as needed during processing. ■ False indicates that Integration Server does not cache partitions when parsing an XML document.
-----------------------	---

Maximum Heap Bytes <code>maximumHeapBytes</code>	<p>Maximum amount of heap space that the parser can allocate to process documents concurrently. Specify a suffix of “k” to indicate kilobytes, “m” to indicate megabytes, “g” to indicate gigabytes, and “%” to indicate a percentage of the heap space. For example, 10k, 10m, 10g, or 10%.</p>
--	--

Maximum Document Bytes <code>maximumDocBytes</code>	<p>Maximum amount of heap space that the parser can allocate to process a single document. Specify a suffix of “k” to indicate kilobytes, “m” to indicate megabytes, “g” to indicate gigabytes, and “%” to indicate a percentage of the heap space. For example, 10k, 10m, 10g, or 10%.</p>
---	---

Use BigMemory <code>useBigMemory</code>	<p>Indicates if BigMemory is used with the enhanced XML parser.</p> <ul style="list-style-type: none"> ■ True indicates that when parsing an XML document using the enhanced XML parser Integration Server moves partitions to BigMemory once the on-heap cache is full. When BigMemory becomes full partitions are move to disk. Integration Server retrieves partitions from disk or BigMemory as needed. ■ False indicates that when parsing an XML document using the enhanced XML parser Integration Server does not use BigMemory to store cached partitions.
---	---

MaximumBig MemoryBytes <code>maximumBigMemoryBytes</code>	<p>Maximum amount of non-heap memory, measured in bytes, to allocate to BigMemory. Specify a suffix of “k” to indicate kilobytes, “m” to indicate megabytes, or “g” to indicate gigabytes. For example, 10k, 10m, or 10g.</p>
---	---

Ports

Asset Type ID	<code>isport</code>
---------------	---------------------

Substitution Values

General values for all ports (email, file polling, FTP, FTPS, HTTP, HTTPS, and quiesce):

Note: Deployer extracts the Ports asset from the package in which the port is configured. Make sure this package resides in the source folder before extracting the asset.

Package Name pkg	The package name you want to associate with the port on the target servers. If the port is used as the quiesce port, the port alias should be associated with either the WmRoot or WmPublic package.
Enable (enable)	Specifies whether the port on the target servers are enabled or disabled after deployment. <ul style="list-style-type: none"> ■ True enables the port on the target servers after deployment. ■ False indicates that you do not want the port enabled on the target servers after deployment. ■ None indicates that you want to use the global value specified by the deployment options on the Deployment Map Properties > Configure Builds by Assets screen and by the Enable Ports After Deployment value in the project map file.
Hosts hostList	A comma delimited list that specifies the hosts allowed or not allowed to connect to the target servers through this port. <p>Note: If the Access Mode is Global, the host list is ignored. If the Access Mode is Allow, the host list represents the hosts that are denied access to the port. If the Access Mode is Deny, the host list represents the hosts that are allowed access to the port.</p>
IP access type hostAccessMode	Specify the access type of host that is allowed to connect to the target servers through this port. <ul style="list-style-type: none"> ■ Allow indicates that you want Integration Server to allow access by default and to deny the hosts specified in the hosts list. ■ Deny indicates that you want Integration Server to deny access by default and allow only hosts specified in the host list. ■ Global indicates that you want to use the global access settings specified on Integration Server. <p>For detailed information about IP access types and controlling IP access, see <i>webMethods Integration Server Administrator's Guide</i>.</p>
Email port:	

Host Name host	The name of the machine on which the POP3 or IMAP server runs.
Port server_port	(Optional.) The port on the e-mail server to which the target servers can connect.
User Name user	A user name that identifies the target servers to the e-mail server.
Password password	The password associated with the user name that identifies the target servers to the e-mail server.
Run services as user runUser	The user name you want the target servers to use when running the service. The target servers run 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.
File polling port:	
Monitor Directory monitorDir	The directory on the target servers that you want to monitor for files.
Working Directory workDir	(Optional.) The directory on the target servers to which the servers 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.
Completion Directory completionDir	(Optional.) The directory on the target servers to which you want files moved when processing is completed in the Monitoring Directory or Working Directory.
Error Directory errorDir	The directory on the target servers to which you want files moved when processing fails.
Enable Clustering	Specifies whether the target servers should allow clustering in the Monitoring Directory.

-
- `clusterEnabled` ■ **Yes** indicates that you want the target servers to allow clustering in the Monitoring Directory.
- **No** indicates that you do not want the target servers to allow clustering in the Monitoring Directory.
-

Run Services as User The user name you want the target servers to use when running the service.

`runUser` The target servers run 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.

Directories are NFS mounted file system 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.

- `NFSDirectories` ■ **No** indicates that you want the listener to call the Java `File.renameTo()` method and move the files from the monitoring directory to the working directory, and from the working directory to the completion and/or error directory. This is the default.
- **Yes** indicates that you want the listener to first call the Java `File.renameTo()` method to move the files from the monitoring directory. If this method fails, the listener will 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.
-

FTP port:

Port The port number you want to use for the FTP port on the target servers. Select a number that is not already in use on the target servers.

`port`

Bind address (Optional.) IP address to which to bind this port. Specify the substitute bind address if the target servers have multiple IP addresses and you want the port to use this specific address.

`bindAddress`

FTPS port:

Port port	The port number you want to use for the FTPS port on the target servers. Select a number that is not already in use on the target servers.
Bind address bindAddress	(Optional.) IP address to which to bind this port. Specify the substitute bind address if the target servers have multiple IP addresses and you want the port to use this specific address.
Secure Clients Only secureclients	Specify whether you want to prevent the FTPS listener from operating with non-secure clients. <ul style="list-style-type: none"> ■ True prevents the FTPS listener from operating with non-secure clients. ■ False allows the FTPS listener to operate with non-secure clients.

HTTP port and HTTPS port:

Port port	The port number you want to use for the HTTP port or HTTPS port on the target servers. Select a number that is not already in use on the target servers.
Bind address bindAddress	(Optional.) IP address to which to bind this port. Specify the substitute bind address if the target servers have multiple IP addresses and you want the port to use this specific address.

Quiesce port:

Quiesce Port quiesceport	The Integration Server port to use as the quiesce port on the target servers. <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: Ensure that the port alias is associated with either the WmRoot or the WmPublic package and is of type HTTP or HTTPS.</p> </div>
------------------------------------	--

Proxy server alias

Asset Type ID isproxyserveralias

Substitution Values

Host Name or IP Address The host name or IP address of the proxy server.

host

Port Number The port on which this proxy server listens for requests.

port

User Name The user name Integration Server must use when accessing this proxy server.

username

Password The password Integration Server must use to access this proxy server.

password

Proxy Server Bypass

Asset Type ID isproxyserverbypass

Substitution Values

Addresses The fully qualified host and domain name of each server to which you want the Integration Server to issue requests directly.

addresses

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*.*

Reliable Messaging Configuration

Asset Type ID isreliablemessaging

Substitution Values

Retransmission Interval The time interval (in milliseconds) for which a reliable messaging source waits for an acknowledgement from the reliable messaging destination before the source retransmits the SOAP message.

retransmission
Interval

Acknowledgement Interval <code>acknowledgementInterval</code>	<p>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.</p>
Exponential Backoff <code>exponentialBackoff</code>	<p>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.</p> <ul style="list-style-type: none"> ■ True specifies that the successive retransmission intervals must be increased 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. ■ False specifies that the retransmission interval is not to be adjusted.
Inactivity Timeout <code>inactivityTimeout</code>	<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.</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 indicate that there is no inactivity timeout limit, set the value of Inactivity Timeout as -1.</p>
Inactivity Timeout Interval <code>inactivityTimeoutMeasure</code>	<p>The unit of measure for the <code>inactivityTimeout</code> property. You can specify the unit of measurement as seconds, minutes, hours, or days. The default is 60 seconds.</p>
Sequence Removal Timeout	<p>The length of time for which a reliable messaging source waits for an acknowledgement from a reliable messaging destination before it terminates the sequence and removes the sequence from the memory.</p>

sequenceRemoval
Timeout

Sequence Removal Timeout Interval Measure The unit of measure for the `sequenceRemovalTimeout` property. You can specify the unit of measurement as seconds, minutes, hours or days. The default is 60 seconds.

sequenceRemoval
TimeoutMeasure

In-Order Delivery Assurance Whether the messages in a sequence must be delivered to a reliable messaging destination in the same order in which they have been sent by the reliable messaging source.

`invokeInOrder`

- **True** specifies that the messages in a sequence must be delivered to the destination in the same order in which they have been sent. The order in which the messages are sent is indicated by the sequence key of each message. This is the default.
- **False** specifies that the delivery of messages in the same order in which they have been sent is not to be enforced.

Maximum Retransmission Count The number of times a reliable messaging source must retransmit a message if an acknowledgement is not received from the reliable messaging destination. The value must be an integer within the range of 1 and 256 (inclusive). The default is 10. To indicate that there is no limit to the number of retransmission attempts, set the value of **Maximum Retransmission Count** as -1.

maximum
Retransmission
Count

Storage Type Specifies whether Integration Server uses the persistent or non-persistent mode to store the reliable messaging sequence information.

`storageType`

- **Non-Persistent** specifies that the reliable messaging sequence information is stored in a non-persistent storage mode. 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** specifies that the reliable messaging sequence information is stored in a persistent storage mode. 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 (Seconds)	Specifies the time interval (in seconds) in which Integration Server sweeps the database to check for timed-out or terminated sequences.
<code>houseKeepingInterval</code>	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 if the <code>sequenceRemovalTimeout</code> interval is completed for the sequence. The default is 20 seconds.

Remote Server Alias

Note: Deployer does not export remote server aliases that use the alias name “local”.

Asset Type ID `isremoteserveralias`

Substitution Values

Host Name	The host name or IP address of the remote server represented by the alias.
------------------	--

`host`

Retry Server	Host name or IP address of the remote server you want the target Integration Server to connect to if the primary remote server is unavailable.
---------------------	--

`retryServer`

Port	The port number that is used by the remote server specified by the alias.
-------------	---

`server_port`

User Name	User name the target server will use to access and invoke services on the remote server.
------------------	--

`user`

Password	Password identified in the user account for User Name .
-----------------	--

`pass`

SAML Token Issuer

Asset Type ID `issamlissuer`

Substitution Values

None.

Scheduled Tasks

Asset Type ID `istask`

Substitution Values

Run as User User name the target servers will use when running the service.

`runAsUser` The target servers run 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.

Cluster Target Node Specifies whether you want the task to run on other target servers in the cluster.

`Target`

- **Any server** indicates that you want the task to run on only one server in the cluster, and it does not matter which one.
- **All servers** indicates that you want the task to run on all servers in the cluster.
- Enter a specific server name in the cluster if the task needs to run on only a specific server.

Note: To use this parameter, the source server must be enabled for clustering and the target servers must belong to a cluster.

Suspend During Deployment Specifies whether you want to suspend the existing task during deployment.

`suspendDuringDeploy`

- **True** indicates that you want to suspend the task during deployment.
- **False** indicates that you want the task to run as scheduled during deployment.

- **None** indicates that you want to use the global value specified on the **Deployment Map Properties > Configure Builds by Assets** screen and by the **Suspend tasks during deployment** value in the project map file. The default value is **None**.

Activate after deployment

Specifies whether you want to activate the existing task after deployment.

`activateAfterDeploy`

- **True** indicates that you want the task to activate after deployment.
- **False** indicates that you want the task to suspend after deployment.
- **None** indicates that you want to use the global value specified on the **Deployment Map Properties > Configure Builds by Assets** screen and by the **Activate tasks after deployment** value in the project map file.

The default value is **True** if the scheduled task is active (not suspended) on the source server. The default is **False** if the scheduled task is suspended on the source server.

SFTP Server Alias

Asset Type ID `issftpserveralias`

Substitution Values

Host Name or IP Address

Host name or IP address of the SFTP server.

`hostName`

Port Number

Port number of the SFTP server. The port number must be within the range of 0 and 65535 (inclusive).

`port`

Host Key Location

Location of the public key file of the SFTP server. Integration Server populates this field with the host key file of the source Integration Server. You can change the value of this field to specify a different host key file for deployment.

`hostKeyLocation`

Note: The public key file must be present on the same machine in which you have installed Integration Server.

SFTP User Alias

Asset Type ID	issftpuseralias
Substitution Values	
User Name userName	User name for the SFTP user account.
Authentication Type authenticationType	The type of authentication that Integration Server uses to authenticate itself to the SFTP server. Client authentication type can be either password or publicKey.
Password password	Password for the specified user to connect to the SFTP server if you are using password authentication.
PassPhrase passPhrase	Passphrase for the private key file of the specified user if you are using public key authentication and if the private key you specified requires a passphrase.
Private Key Location privateKeyFileLocation	The location of the private key file of the specified SFTP user if you are using public key authentication. Integration Server populates this field with the private key file of the source Integration Server. You can change the value of this field to specify a different private key file for deployment.
Maximum Retries maximumRetries	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 connectionTimeout	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.
Session Timeout sessionTimeout	The number of minutes you want Integration Server to wait before terminating an idle session. The default is 10 minutes.

Proxy Alias proxyAlias	Proxy alias through which the request is to be routed.
Compression compression	<p>Specifies whether or not to compress the data to reduce the amount of data that is transmitted. Integration Server supports compression using the compression algorithm <code>zlib</code>.</p> <ul style="list-style-type: none"> ■ zlib indicates that you want to compress the data that is transmitted between the SFTP server and Integration Server. ■ none indicates that you do not want to compress the data. <p>Note: You can use compression only if the SFTP server that you are connecting to supports compression.</p>
Compression Level compressionLevel	The compression level to use if you specified the compression algorithm <code>zlib</code> for the Compression property. The minimum allowed value is 1 (fast, less compression). The maximum allowed value is 6 (slow, most compression). The default is 6.
SFTP Server Alias sftpServerAlias	The alias of the SFTP server to which you want the user specified using the User Name property to connect.
Truststore Alias	
Asset Type ID	istruststorealias
Substitution Values	
Location ksLocation	Full path to the Integration Server's client truststore file.
Password ksPassword	Password associated with this alias that is used to protect the contents of the truststore.
Use Source Truststore useSourceTruststore	<p>Specifies whether the target Integration Server uses the truststore file from the source Integration Server.</p> <ul style="list-style-type: none"> ■ True indicates that the target Integration Server uses the truststore file from the source Integration Server. This is the default.

- **False** indicates that the target Integration Server will use a different truststore file than the source Integration Server.

Note: This configuration value does not correspond to a field or property in Integration Server.

Users

Asset Type ID `isuser`

Substitution Values

None.

Integration Cloud Accounts

Asset Type ID `iswmccloudaccount`

Substitution Values

Stage The Integration Cloud stage from which the on-premise Integration Server listens for messages. This value refers to the stage created by the user on Integration Cloud.
`stage`

Note: Regardless of what the stages are named on Integration Cloud, the stages are deployed using the following names:

- `stage00stage01stage02stage99`

Allowed On-Premise Hosts Comma-separated list of the on-premise Integration Servers that can listen for messages for a particular account.

`allowedOnPremiseHosts`

Integration Cloud Applications

Asset Type ID `iswmccloudapplication`

Substitution Values

None.

Integration Cloud Settings

Asset Type ID iswmcloudsettings

Substitution Values

User Name The user name the on-premise Integration Server must use to access Integration Cloud.
user

Password The password the on-premise Integration Server must use to access Integration Cloud.
password

Integration Cloud URL The URL of the Integration Cloud server.
wmCloudURL

Web Service Endpoint Alias

Asset Type ID iswebserviceendpointalias

Substitution Values**Transport Properties:**

Host Name Host name or IP address of the server on which the web service resides. (Provider and consumer web service endpoint alias only.)
host

Port Number Active HTTP or HTTPS type listener port defined on the host server or proxy server. (Provider and consumer web service endpoint alias only.)
port

User Name User name used to authenticate the consumer at the HTTP or HTTPS transport level on the host server. (Consumer and message addressing web service endpoint alias only.)
transportUser

Password The password used to authenticate the consumer on the host server. (Consumer and message addressing web service endpoint alias only.)
transport
Password

Authentication Type	The type of authentication to authenticate the consumer at the HTTP or HTTPS transport level on the host server. (Consumer and message addressing web service endpoint alias only.)
transportAuthType	<ul style="list-style-type: none"> ■ Basic indicates that basic authentication (user name and password) is used to authenticate the consumer. ■ Digest indicates that password digest is used to authenticate the consumer.

Message Properties:

User Name	The user name to include with the UsernameToken. (Consumer web service endpoint alias only.)
messageUser	

Password	The password to include with the UsernameToken (must be plain text). (Consumer web service endpoint alias only.)
messagePassword	

Partner's Certificate	Path and file name of the file containing the provider's certificate. (Consumer and message addressing web service endpoint alias only)
partnerCertificateFileName	

Use Source Partner's Certificate	Specifies whether the target Integration Server uses the partner certificate file from the source.
useSourcePartnerCertificate	<ul style="list-style-type: none"> ■ True indicates that the target server uses the partner certificate file from the source. During deployment, Deployer copies the partner's certificate from the source machine to the location on the destination machine specified in Partner's Certificate. This is the default value. ■ False indicates that the target server will use a different partner certificate file than the source the source partner.

Note: This configuration value does not correspond to a field or property in Integration Server.

Kerberos Credentials:

JAAS Context	The custom JAAS context used for Kerberos authentication.
jaasContext	

Client Principal	The name of the client principal to use for Kerberos authentication.
<code>clientPrincipal</code>	

Service Principal Name	The service that the Kerberos client wants to access. This can be obtained from the WSDL document published by the provider of Kerberos service.
<code>servicePrincipal</code>	

Service Principal Name Form	The format in which you want to specify the principal name of the service that is registered with the principal database, namely, host-based or username.
<code>servicePrincipalForm</code>	

Message Addressing Properties:

To	Endpoint reference to which the SOAP message is sent. (Consumer web service endpoint alias only.)
<code>toMsgAddr</code>	

From	Endpoint reference containing the source of the SOAP message. (Consumer and message addressing web service endpoint aliases only.)
<code>fromMsgAddr</code>	

ReplyTo	Endpoint reference containing the destination address of the response (reply) message. (Consumer and message addressing web service endpoint aliases only.)
<code>replyToMsgAddr</code>	

FaultTo	Endpoint reference containing the address to which the SOAP fault messages are routed. (Consumer and message addressing web service endpoint aliases only.)
<code>faultToMsgAddr</code>	

Reliable Messaging Properties

Note: The reliable messaging properties apply only to consumer and provider web service endpoint aliases of transport type HTTP and HTTPS.

Enable	Whether Integration Server uses the reliable messaging properties specific to the web service endpoint alias or the server-level reliable messaging properties that applies to all web service endpoints in the server.
<code>enable</code>	

- **True** specifies that Integration Server uses the reliable messaging properties specific to the web service endpoint alias.
- **False** specifies that Integration Server uses the server-level reliable messaging properties.

Note: Rest of the reliable messaging properties specific to a web service endpoint alias are the same as the server-level reliable messaging properties. For more information about reliable messaging properties, see [Reliable Messaging Configuration](#) under "[Integration Server Assets](#)" on page 257.

Web Service Policy

Asset Type ID `iswspolicy`

Substitution Values

None.

Integration Server Administrative Asset Dependencies

The following table lists the dependent assets and the reference assets that you must include in a deployment set before you can deploy the dependent asset:

Asset	Dependencies
JDBC pool alias	JDBC driver alias
JDBC functional alias	JDBC driver alias
Client certificate	User <p>Note: Default users (for example, Administrator) are not listed as dependencies.</p>
LDAP configuration	Group <p>Note: Default groups are not listed as dependencies.</p>

Asset	Dependencies
Remote server alias	Keystore alias, ACL Note: Default ACLs (for example, Anonymous) are not listed, and the local remote server alias is not extracted.
SAML token issuer	Truststore alias
URL alias	IS package
Integration Cloud Accounts	User name identified by the Run As User field.
Integration Cloud Applications	Integration Cloud accounts and Integration Server package assets
Web service endpoint alias	Keystore alias, truststore alias, JMS trigger name (provider JMS), proxy alias (consumer HTTP and HTTPS), keystore alias (consumer HTTPS), JNDI alias (consumer JMS), JMS alias (consumer JMS), web service endpoint alias (provider HTTP, HTTPS, and JMS)

Integration Server Package Assets

This section describes the following:

- [Adding Package Assets to the Source Directory](#)
- [Global Values for Integration Server Package Assets and Composites](#)
- [Individual Values for Integration Server Package Assets](#)

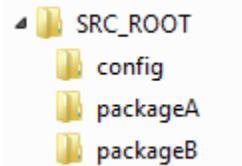
About Integration Server Packages

Integration Server packages can be deployed as either composites or assets with a type ID of ispackage.

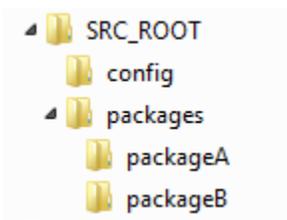
Adding Package Assets to the Source Directory

To include package assets in the composite for deployment, you must manually copy or check in the *Integration Server_directory*\instances*instance_name* \packages folder to the source directory. There are two ways to add package assets to the source directory:

- **Build package composites along with other administrative assets.** If you want to generate composites for all packages in the source directory and package-specific administrative assets, Software AG recommends that you structure the source directory as either:



Or:

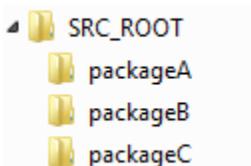


Note: In order to retain ACL information for the package assets, you must add ACL configuration files to the config folder.

In the above example, you would define SRC_ROOT as the value of the build.source.dir property in the build.properties file. For more information about the build.properties file, see ["Setting Build Properties" on page 32](#).

When run, the build script creates the following:

- A composite named *package_name*.zip for each package included in the source directory, where *package_name* is the name of the package (with a composite type ID of ispackage). For this example, the files would be named packageA.zip and packageB.zip.
 - A composite named isconfiguration.zip (with a composite type ID of isconfiguration) that contains the administrative assets contained in the config directory.
- **Build package composites separate from other administrative assets.** If your source directory contains several packages and you want to generate composites from only a select number of those packages, you can use the build.source.project.dir property to specify only those packages you want to include. For example, in the following source directory there are three packages: packageA, packageB, and packageC:



You can generate a composite that contains only packageA and packageB by setting the value of the `build.source.project.dir` property to:

```
SRC_ROOT/packageA;SRC_ROOT/packageB
```

In this example, since the config directory is not located in `SRC_ROOT`, you must specify the location of config directory in the `is.acdl.config.dir` property of the `build.properties` file.

When run, the build script creates a composite named `package_name.zip` for each package defined for `build.source.project.dir`, where `package_name` is the name of the package (with a composite type ID of `ispackage`).

Using the above example, `build.source.project.dir` is set to `"SRC_ROOT/packageA;SRC_ROOT/packageB"`. When the build script runs, it will generate composites for packageA and packageB. Since packageC is not defined for `build.source.project.dir`, the build script ignores it. Since the packages are named "packageA" and "packageB" in the source directory, the build script names the composites `packageA.zip` and `packageB.zip`.

For more information about setting properties in the `build.properties` file, see ["Setting Build Properties" on page 32](#). For more information about building composites for repository-based deployment, see ["Building Composites for Repository-Based Deployment" on page 29](#).

Global Values for Integration Server Package Assets and Composites

The following table lists the global values you can export for Integration Server packages.

Value	Description
Activate Package on Install (activatePkgOnInstall)	Specifies whether you want the Integration Server to activate the package immediately upon installation. <ul style="list-style-type: none"> ■ True indicates that you want the server to activate the package after it is installed. ■ False indicates that you do not want the server to activate the package after it is installed. The default value is True .
Archive Package on Install (archivePkgOnInstall)	Specifies whether you want Integration Server to archive the package automatically after installation. <ul style="list-style-type: none"> ■ True indicates that you want the server to archive the package automatically after it is installed. ■ False indicates that you do not want the server to archive the package after it is installed.

Value	Description
	The default value is True .
Compile Package on Install (compilePackage)	<p>Specifies whether you want Integration Server to compile the package automatically after installation.</p> <ul style="list-style-type: none"> ■ True indicates that you want the server to compile the package automatically after it is installed. ■ False indicates that you do not want the server to compile the package after it is installed. <p>The default value is True.</p>
Disallow Active Package Install (disallowActivePackageInstall)	<p>Specifies whether you want to prevent deployment if the package being deployed is in an active state on the target Integration Server.</p> <ul style="list-style-type: none"> ■ True indicates that you want to prevent the server from continuing deployment when the package being deployed is in an active state on the target Integration Server. ■ False indicates that you want the server to continue deployment regardless of whether the package is active on the target Integration Server. <p>The default value is False.</p>
Fragment Package on Install (fragPackage)	<p>Specifies whether you want Integration Server to perform a fragmentation step during the compilation of a package. The fragmentation step creates new node.ndf files for any Java services contained in the package. Omitting this step retains the node.ndf files that were copied from the source server, thereby preserving custom settings defined in those files.</p> <ul style="list-style-type: none"> ■ True indicates that you want to allow the server to perform the fragmentation step and overwrite a package's node.ndf files when the package is compiled on the target server. ■ False indicates that you want the server to omit the fragmentation step and retain the node.ndf files that were copied from the source server when the package is compiled on the target server. <p>The default value is True.</p>
Package Execution Check	<p>Specifies the length of time (in milliseconds) Deployer should wait if a service contained in the package being deployed is being executed on the target Integration Server. If this time expires and a service is still being executed, Deployer terminates</p>

Value	Description
(packageExecutionCheck)	<p>the deployment job. The default value for this parameter is 0, which disables this feature.</p> <p>Note: In some cases, this parameter can override <code>disallowActivePackageInstall</code>. For example, if <code>disallowActivePackageInstall</code> is set to False and <code>packageExecutionCheck</code> is set to a value other than 0, Integration Server can terminate the deployment job even though <code>disallowActivePackageInstall</code> would otherwise allow deployment to succeed.</p>
<p>Suspend Triggers During Deployment</p> <p>(suspendTriggersDuringDeploy)</p>	<p>Specifies whether you want Integration Server to suspend existing triggers before updating them with the deployment.</p> <ul style="list-style-type: none"> ■ True indicates that you want the server to suspend existing triggers before deployment. ■ False indicates that you do not want the server to suspend the triggers. <p>The default value is False.</p>
<p>Synchronize Document Types To Broker During Deployment</p> <p>(syncDocTypesToBroker)</p>	<p>Specifies whether Integration Server should synchronize the publishable document types in the source package with document types on the Brokers that are connected to the target Integration Server.</p> <ul style="list-style-type: none"> ■ True indicates that you want Integration Server to synchronize publishable document types in the target server with the connected Brokers during deployment. ■ False indicates that you do not want Integration Server to synchronize publishable document types in the target server with the connected Brokers during deployment. <p>The default value is True.</p> <p>Note: The target Integration Server must be connected to a Broker at deployment time for synchronization to occur. If the connected Broker is not available, publishable document types are not synchronized for the Integration Server to which the Broker is connected. Deployer writes a message to that effect to the deployment report.</p>
<p>Clear ACLs after deployment</p> <p>(clearACL)</p>	<p>Specifies whether to reset ACLs on the assets during deployment.</p> <ul style="list-style-type: none"> ■ True indicates that Integration Server will remove any ACLs that are set for the assets.

Value	Description
	<ul style="list-style-type: none"> ■ False (the default) indicates that Integration Server should not remove any existing ACLs for the assets.

Individual Values for Integration Server Package Assets

Integration Server supports the following Integration Server package assets for deployment on a project-by-project basis. The values presented in the table are deployed as assets within the ispackage composite. None of the assets support substitutions.

Asset	Asset Type ID	Notes
BlazeRule services	blazeruleservice, isfile	Blaze rules consist of an Integration Server Java service and two files in the config directory. Integration Server handles the Integration Server Java service as a normal Integration Server Java service, and handles the configuration files as package files. The Integration Server service does <i>not</i> state a dependency on the config files, so assets that depend on a Blaze rule service will need to have dependencies on both the Integration Server service and the configuration files.
webMethods messaging triggers	istrigger	None.
C services	iscservice	None.
Document types	isdocumenttype	None.
E-forms	isdocumenttype, isfile	E-forms consist of an IS document and a configuration file. The IS document is handled as a normal IS document asset. Integration Server handles the configuration file as a package file. The IS document does <i>not</i> state a dependency on the config files (they are not associated in Integration Server), so assets that depend on an e-form will need to have dependencies on both the IS document and the configuration file.

Asset	Asset Type ID	Notes
Flat file dictionaries	ffdictionary	None.
Flat file schemas	ffschema	None.
Flow services	isflowservice	None.
IS schemas	isschema	None.
Java services	isjavaservice	None.
JMS triggers	isjmstrigger	None.
Package files	isfile	None.
Package folders	isfolder	None.
PRT fragments	isfile	Handled as a package file.
Specifications	isspecification	None.
URL aliases	isurlalias	<p>The URL Alias asset refers to the URL aliases created for the server or a package. The URL Alias asset does not refer to the URL aliases created for services in Designer or Developer. URL aliases for services are deployed along with the service when a package is deployed.</p> <p>The URL aliases for a server are deployed to the WmRoot package.</p>
Web service connectors	iswsconnector	None.
Web service descriptor consumers	iswsdconsumer	None.

Asset	Asset Type ID	Notes
Web service descriptor providers	iswsdprovider	None.
XSLT services	isxslt-service	None.

Adapter Runtime and .NET Service Assets

This section describes the adapter runtime (ART) assets and the .NET asset that Integration Server supports for exporting, and their dependencies.

Adding Adapter Runtime and .NET Service Assets to the Source Directory

To include adapter runtime and .NET service assets in the composite for deployment, you must manually copy or check in the *Integration Server_directory*\instances*instance_name* \packages folder to the source directory.

The build script creates a composite named *package_name*.zip, where *package_name* is the name of the package. For example, if the package name is "adapter_service", the composite name is adapter_service.zip. For more information about building composites for repository-based deployment, see "[Building Composites for Repository-Based Deployment](#)" on page 29.

Adapter Runtime Assets

The following table lists the adapter runtime assets and values for all WmART based adapters that you export and deploy using Integration Server:

Asset	Asset Type ID	Substitution Values
Adapter connections	artconnection	A list of connection management properties, and connection properties based on the adapter. The connection properties are a dynamic set and depend on the adapter implementation.

Note: Before you deploy an adapter connection, you must set the password for the connection. For more information, see the installation and user's guide for that adapter.

Asset	Asset Type ID	Substitution Values
		<p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment. Valid values:</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>
Adapter services	artservice	None.
Adapter listeners	artlistener	<p>Retry Limit (<code>retryLimit</code>). The number of times that the system attempts to start the listener if the initial attempt fails. In particular, this field specifies how many times to retry the <code>listenerStartup</code> method before issuing an adapter connection exception. The default is 5. A value of 0 indicates that the system will make a single attempt.</p> <p>Retry Backoff Timeout (<code>retryBackoffTimeout</code>). The number of seconds the system waits between each attempt to start the listener. The default is 10. The value of this field is ignored if Retry Limit is set to 0.</p> <p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment. Valid values:</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>
Adapter listener notifications	artlistener noification	<p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment. Valid values:</p>

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>
Adapter polling notifications	artpolling notification	<p>Interval (<code>notificationInterval</code>). The time of the polling interval in seconds.</p> <hr/> <p>Overlap (<code>notificationOverlap</code>). The time the scheduled interval time you set in the Interval field will begin.</p> <ul style="list-style-type: none"> ■ True indicates that the next execution of a scheduled notification does not wait for the current execution to end and they overlap. ■ False indicates that the executions of a scheduled notification do not overlap. <hr/> <p>Immediate (<code>notificationImmediate</code>). Whether to enable or disable polling immediately.</p> <ul style="list-style-type: none"> ■ True enables polling immediately. ■ False disables polling immediately. <hr/> <p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment.</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>

Adapter Runtime Asset Dependencies

The following table lists the dependent assets and the reference assets that you must include in a deployment set before you can deploy the dependent asset:

Asset	Dependency
Adapter Connections	None
Adapter Services	Adapter Connection
Adapter Listeners	Adapter Connection
Adapter Polling Notifications	Adapter Connection, IS Document Type
Adapter Listener Notifications	Adapter Listener, IS Document Type

.NET Asset

The following table lists the .NET asset that you export and deploy using Integration Server and its values:

Note: The .NET service asset is not a dependent asset. However, ensure that the values in the **Assembly Path** and **Assembly Name** fields that you provide for the target Integration Server during variable substitution are valid.

Asset	Asset Type ID	Substitution Values
.NET Services	dotnetservice	<p>Assembly Path (<code>assemblyPath</code>). The location of the directory that holds the .NET assembly in which the method called by the .NET services resides.</p> <hr/> <p>Assembly Name (<code>assemblyName</code>). The name of the .NET assembly in which the method called by the .NET services resides.</p>

Mobile Support Assets

Using Deployer, you can deploy the following webMethods Mobile Support assets:

- Mobile sync component configuration
- Name and version of the mobile application associated with a mobile sync component

These assets are deployed as assets within the isconfiguration composite when the WmMobileSupport package is installed. The following table lists the dependencies for these assets.

These assets should be deployed together. For details about adding these assets to the source directory, see ["Adding Administrative Assets to the Source Directory" on page 258](#).

Asset	Asset Type ID	Dependencies
Mobile applications used by Mobile Support	mobileAppSetting	None.
Mobile sync components for Mobile Support	MSCConfigSetting	Ensure that the business document type and the download and upload services specified for the mobile sync components are present on the target server.

My webMethods Server Assets

This section lists the My webMethods Server (including Task Engine) asset types and their dependencies.

The following table lists the dependencies and properties that you can substitute when using Deployer to deploy assets to a My webMethods Server.

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Access privilege	AccessPrivilege	The rights of a user, group, or role to view applications and features in the Navigation

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>panel and to access pages associated with them. See Functional privileges.</p> <p>Dependencies: The user, group, or role specified by the access privilege.</p>
Business calendar	Calendar	A global calendar used in My webMethods Server (for example, a US holidays calendar).
CAF application	CAFApp	<p>A Composite Application Framework application.</p> <p>Substitution variables: Any env-properties or configuration properties from the application's web.xml file (for example, properties dynamically added when you create a web connector for your portlet).</p>
CAF runtime configuration	RuntimeConfig	<p>The runtime configuration for a Composite Application Framework application, as modified in the CAF Runtime Administration page.</p> <p>Substitution variables: Any env-properties or configuration properties from the application's web.xml file (for example, properties dynamically added when you create a web connector for your portlet).</p>
Certificate	Certificate	A digital certificate associated with an My webMethods Server user.
Component	Component	<p>A legacy portlet or Dynamic Business Object (DBO).</p> <p>Dependencies: Any portlets or DBOs referenced by the xmlImport.xml file of the Component.</p>
Data source	Datasource	<p>An external data source configuration (for example, a connection to an external Oracle database).</p> <p>Substitution variables: The user name, password, and URL for the data source.</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Directory service	Directory	<p>Configuration of LDAP or Database directory service from My webMethods Server.</p> <p>Dependencies:</p> <ul style="list-style-type: none"> ■ LDAP directory service - None ■ Database directory service - Data Source <p>Substitution variables: The user name, password, and LDAP URL for the LDAP directory service only. There are no substitution variables for the database directory service.</p>
Functional privilege	FuncPrivilege	<p>The rights of a user, group, or role to make changes within an application or feature, such as to create and modify a workspace. You can export all of the functional privileges associated with a specific user, group, or role. See Access Privileges.</p> <p>Dependencies: Users, groups, or roles to which functional privileges are granted.</p>
Group	Group	<p>A collection of users and other groups. Groups are defined and stored in an internal system or external directory service.</p> <p>Dependencies: The external directory service where the groups are defined. Only groups from internal "System" directory service can be deployed. Groups from LDAP and Database directory come implicitly from the directory itself.</p>
Locale rule	Rule.locale	<p>A rule configuration that defines the locale (language and country code) to use when locale information is not specified in the user profile.</p> <p>Dependencies: Any users, groups, or roles referenced by the rule logic.</p>
Login page rule	Rule.login	<p>A login page rule configuration that defines the login page to be used.</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>Dependencies: Any users, groups, or roles referenced by the rule logic of the login page that is the target of the rule.</p>
Page/Folder	Folder	<p>The layout and content of a folder or page.</p> <p>Dependencies: The users, groups, and roles referenced in the folder access control. In addition, folders and pages have a dependency on the portlets and Composite Application Framework applications included in the page.</p>
Portlet	Portlet	<p>A portlet is a mini-application or a piece of functionality that can be part of a Composite Application Framework application or can reside independently on a page on the server.</p>
Rendering rule	Rule.render	<p>The user interface formatting capabilities assigned to specific server objects by defining rendering rules. Renderer rules determine the specific renderer to use.</p>
Role	Role	<p>A collection of users assigned to a specific role defined for any directory service.</p> <p>Dependencies: Users, groups, or roles contained within the role.</p>
Saved search	Search	<p>A saved query associated with a particular search page within My webMethods Server (for example, Task List Management or My Inbox). A Saved Search asset includes all saved searches for a particular search page (not individual searches).</p>
Security realm	Realm	<p>A way to apply an access control list to a list of objects. You can organize Security Realms into containers.</p> <p>Dependencies: Users, groups, or roles, and the pages and folders controlled by the security realm.</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Shell	Shell	<p>An installable component used to display the header, footer, and title bars for pages.</p> <p>Dependencies: Any custom portlets referenced by the shell section pages.</p>
Shell rule	Rule.shell	<p>A rule configuration that determines what shell should be displayed for each page.</p> <p>Dependencies: Any users, groups, or roles referenced by the rule logic of the shell that is the target of the shell rule.</p>
Skin	Skin	<p>An installable component that defines the look and feel of the rendered page. A skin modifies the images, fonts, colors, and other subtle style aspects of HTML content, without functionally modifying the HTML content.</p>
Skin rule	Rule.skin	<p>A rule configuration that determines what skin should be displayed.</p> <p>Dependencies: Any users, groups, or roles referenced by the rule logic of the skin that is the target of the skin rule.</p>
Start page rule	Rule.startpage	<p>A rule that determines which page is displayed after a user logs into the server.</p> <p>Dependencies: Any users, groups, or roles referenced by the rule logic of the page or folder that is the target of the rule.</p>
Task	Task	<p>Represents human activity for a BPM ProcessModel. Tasks are composed of Task Rules and portlets that implement UIs of the task.</p> <p>Dependencies:</p> <ul style="list-style-type: none"> ■ Task Rule ■ Portlets ■ Business Rules (for assignments)

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Task rule	Rule.task	A rule within a task type to specify task assignments and behaviors. Dependencies: Any users, groups, or roles referenced by the rule logic.
User	User	An individual listed in the internal “system” directory service and all profile and preference attributes.

Optimize Assets

webMethods Optimize enables you to export assets such as dimensions, event maps, dimension hierarchies, data filters, process configurations, intelligent links, or rules to Deployer. Using Deployer, you can deploy your Optimize assets to another Optimize Analytic Engine and share the assets that you created in Optimize with other applications.

This section lists the following:

- Asset types that Optimize supports for exporting
- Asset dependencies

Asset	Asset Type ID	Dependencies
Custom trees	OptimizeCustomTree	KPIs, Process models
Data filters	OptimizeDataFilter	Dimensions
Dimensions	OptimizeDimension	Intelligent links
Dimension hierarchies	OptimizeHierarchy	Dimensions
Event maps	OptimizeEventMap	Dimensions, Intelligent links
Intelligent links	OptimizeILink	None
KPIs	OptimizeKpi	Dimension hierarchies, event maps

Asset	Asset Type ID	Dependencies
		You cannot deploy individual KPIs. Instead, Deployer deploys all KPIs for a selected event map.
Process configurations	OptimizeProcessConfig	Process models
Rules	OptimizeRule	Event maps, Data filters, KPIs

Trading Networks Assets

This section lists the following:

- Trading Networks asset types you can deploy
- Asset dependencies

Note: Substitution values are case sensitive.

Asset	Asset Type ID	Dependencies, Substitution Variables
Archive schedules	archiveschedule	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Partner profiles ■ BizDocType <p>Substitution Variables:</p> <p>Archive Schedule Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ suspended
Binary types	binarytype	None
Contact types	contacttype	None
Data permissions (DLSs)	dls	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Partner profiles ■ Document types

Asset	Asset Type ID	Dependencies, Substitution Variables
		<ul style="list-style-type: none"> ■ Processing rules ■ My webMethods roles
Document attributes	documentattribute	<p>Dependencies: None</p> <p>Substitution Variables:</p> <p>Document Attribute Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ disabled
Document types	documenttype	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Document attributes ■ ESB services <p>Substitution Variables:</p> <p>Document Type Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ disabled
External ID types	externalidtype	None
Field definitions	fielddefinition	Dependencies: Field groups
Field groups	fieldgroup	None
General functional permissions	functionalpermission	Dependencies: My webMethods roles
Partner profiles	partner	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Contact types ■ External ID types ■ Profile groups ■ Field definitions

Asset	Asset Type ID	Dependencies, Substitution Variables
		<ul style="list-style-type: none"> ■ ESB services <p>Substitution Variables:</p> <ul style="list-style-type: none"> ■ Delivery Method Name: Host ■ Delivery Method Name: Port ■ Delivery Method Name: Username ■ Delivery Method Name: Password ■ Partner Profile Status <ul style="list-style-type: none"> ■ active ■ inactive
Processing rules	processingrule	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ ESB services ■ Partner profiles ■ Document attributes ■ Document types ■ Profile groups <p>Substitution Variables:</p> <p>Processing Rule Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ disabled
Profile groups	profilegroup	None
Queues	queue	<p>Dependencies: ESB services</p> <p>Substitution Variables:</p> <p>Queue Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ disabled ■ draining ■ suspended

Asset	Asset Type ID	Dependencies, Substitution Variables
Trading partner agreements (TPAs)	tpa	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Partner profiles ■ IS document types ■ ESB services <p>Substitution Variables:</p> <p>TPA Status - The values are:</p> <ul style="list-style-type: none"> ■ agreed ■ proposed ■ disabled

Trading Networks logs the deployment activities as activity logs in My webMethods for easy tracking and monitoring of deployment activities.

Universal Messaging Assets

There are some assets which can be modified if they already exists and some which cannot. User will then have to use the Deployer's deletion set feature to delete the asset if it already exists before deploy operation.

This section lists the following:

- Universal Messaging assets you can deploy
- Dependencies and substitution variables (if applicable) of each asset
- Considerations when deploying Universal Messaging.

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Realm ACLs	RealmAcl	<p>Realm ACL assets can consist of either ACLs in <i>user@host</i> format or groups that contain sets of ACLs.</p> <p>Deployer always replaces target assets.</p> <p>Dependencies: Security groups</p>
Security groups	RealmSecurityGroup	If a security group asset of the same name exists on the target

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>server, you must delete the security group from the target server before deploying. For more information about deleting assets from target servers, see "Defining a Deletion Set" on page 127.</p>
Realm schedules	RealmSchedule	<p>If a realm schedule asset of the same name exists on the target server, you must delete the realm schedule from the target server before deploying. For more information about deleting assets from target servers, see "Defining a Deletion Set" on page 127.</p>
Realm configurations	RealmConfig	<p>Deployer always replaces target assets.</p>
Channels	Channel	<p>If the asset exists on the target server, only the ACL can be modified during deployment. If you want to modify additional properties, you must delete the channel asset on the target server before deploying. For more information about deleting assets from target servers, see "Defining a Deletion Set" on page 127.</p>
Channel joins	ChannelJoin	<p>Deployer always replaces target assets.</p> <p>Dependencies: Source channel and target queue/target channel</p>
Queues	UMQueue	<p>If the asset exists on the target server, only the ACL can be modified during deployment. If you want to modify additional properties, you must delete the queue asset on the target server before deploying. For more information about deleting assets</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>from target servers, see "Defining a Deletion Set" on page 127.</p>
Interfaces	Interface	<p>The interface asset can contain four types of interfaces: nsp, nsps, nhp, and nhps. For all of the interfaces, Deployer will modify the VIA list on the target server. The nhp and nhps interfaces can contain plugins, which can also be modified.</p> <p>If an interface asset of the same name exists on the target server, you must delete the interface from the target server before deploying. For more information about deleting assets from target servers, see "Defining a Deletion Set" on page 127.</p> <p>Substitution values: Available for nsps and nhps interfaces only:</p> <ul style="list-style-type: none"> ■ Keystore ■ Keystore Password ■ Truststore ■ Truststore Password ■ Private Key Password
Data groups	DataGroup	<p>Deployer modifies publishers and nested groups on the target server.</p> <p>If a data group asset of the same name exists on the target server, you must delete the data group from the target server before deploying. For more information about deleting assets from target servers, see "Defining a Deletion Set" on page 127.</p> <p>Dependencies: Nested groups</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Clusters	Cluster	Deployer always replaces target assets.
Connection Factories	ConnectionFactory	If the asset exists on the target server, you cannot replace or modify the asset.
Topic Connection Factories	TopicConnectionFactory	If you want to replace the asset in the target server, you must first delete the existing asset in the target server. For more information about deleting assets from target servers, see "Defining a Deletion Set" on page 127 .
Queue Connection Factories	QueueConnectionFactory	
XA Connection Factories	XAConnectionFactory	
JNDI Topics	JNDITopic	
JNDI Queues	JNDIQueue	

Other Assets

The following table describes additional user-created assets that you can include in deployment projects. It includes:

- Assets you can deploy
- Asset dependencies

Asset	Asset Type ID	Dependencies
BAM process models	bamprocess	None.