Migrating webMethods Broker to Software AG Universal Messaging

Version 10.5

October 2019

Document ID: WEBM-MG-105-20210311

Table of Contents

About this Guide

This migration guide contains information about preparing for migration, running migration scripts, and post-migration configuration.

Important:
Broker is deprecated.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com.

Software AG Empower Product Support Website

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

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

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

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

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

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

Software AG TECHcommunity

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

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

Data Protection

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

1 Introduction to Universal Messaging

Overview of Universal Messaging

Universal Messaging is a fast, reliable, scalable, and flexible message-oriented middleware that provides messaging functionality such as clustering, scheduling, and interface plug-ins, with standard support for the messaging paradigms of publish/subscribe, message queues, and P2P, as well as support for JMS.

Universal Messaging provides high-speed, cost-effective, real-time communication solution for enterprise, web, and mobile applications. For additional information about Universal Messaging, see the Universal Messaging webhelp on the Software AG documentation website at http://documentation.softwareag.com.

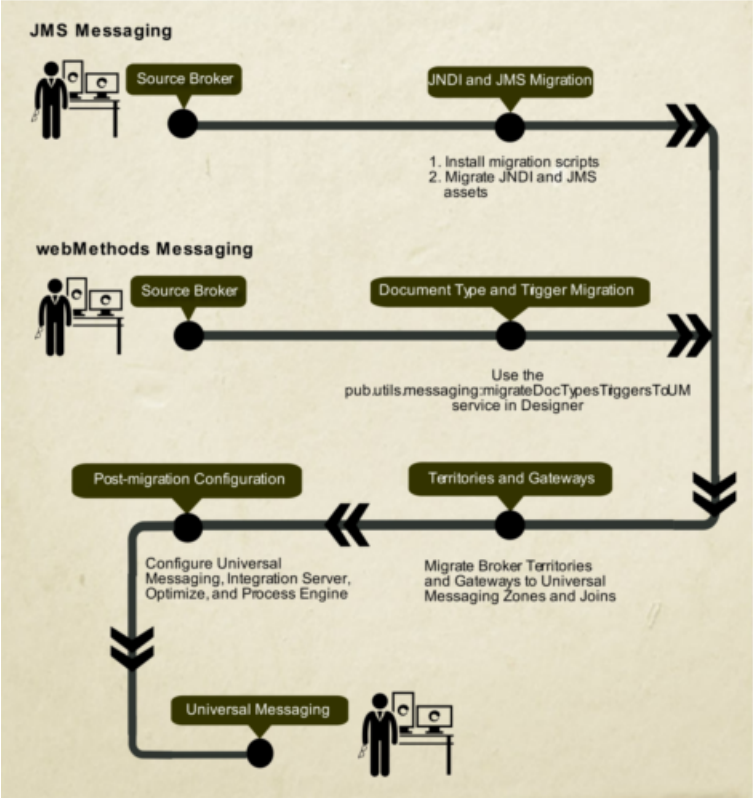
Advantages of Universal Messaging

Universal Messaging provides similar or better than the messaging capabilities that are available in Broker. Universal Messaging has the following advantages over Broker:

- Multiple UM servers per IS for webMethods Messaging (native publish-subscribe)
- Active/active clustering
- Enhanced Command Central support
- High-speed JMS messaging
- Large message volume handling
- Multicast delivery (enabling low latency)
- Concurrent connections scaling (above 100K)
- .NET, C++, and Java client support
- Mobile application support for Apple iOS, Android
- Web application support for HTML5 web sockets, JavaScript, and Java
- Unicast, Multicast, IPC, JMS, MQTT, and HTML5 AMQP protocol support
- Group-based server administration
- API support for merging multiple partial events and message replay
- Channel capacity management
- Message service quality reconfigurability

The following information graphic shows the workflow for migrating from Broker to Universal Messaging.

Important:
Before you start the migration from webMethods Broker to Universal Messaging, refer "Pre-migration Considerations" on page 18 section.



Understanding Universal Messaging Features

Universal Messaging Feature Comparison

The following table contains a comparison of standard features supported in Broker and Universal Messaging:

Standard feature	Broker	Universal Messaging
JMS	O	O
MQTT		O
AMQP		O
C#	O	O
C	O	
C++		O
Java	O	O
JavaScript		O

Standard feature	Broker	Universal Messaging
Active/active clustering		O
Active/passive clustering	O	O
Docker		O
iOS		O
Andriod		O

The following table contains a comparison of features in Broker and different versions of Universal Messaging. O indicates that the feature is available.

Feature	Broker	Universal Messaging 9.12	Universal Messaging 10.1	Universal Messaging 10.3
pub.publish:publish	O	O		
Client side queue (CSQ)	O	O		
publishAndWait/reply	O	O		
pub.publish:deliver	O	O		
Native triggers	O	O		
Local trigger filters	O	O		
Server-side trigger filters	O	O		
Trigger joins	O	O		
Multiple conditions	O	O		
Duplicate detection	O	O		
Multiple servers for webMethods Messaging		O		
SSL/basic auth connections	O	O		
Automatic subscriber interest propagation	O	O		
JCA adapter	O	O		

Feature	Broker	Universal Messaging 9.12	Universal Messaging 10.1	Universal Messaging 10.3
XA	O			
Document auditing	O			
Concurrent JMS trigger	O	O		
Concurrent Native trigger	O	O		
Serial JMS trigger	O		O	
Serial Native trigger	O	O		
Horizontal scaling JMS publish	O			O
Horizontal scaling native publish				O
Horizontal scaling JMS subscribe	O			O
Horizontal scaling Native subscribe				O

The following table explains the types of durable subscribers:

Type	Notes	wM Messaging Triggers	JMS Triggers	Purge allowed
Shared	Allows multiple consumers, where each messages reaches one consumer. More efficient internal implementation.	Concurrent (10.0+)	Shared durable Connection Factory (10.0+)	Yes
Exclusive	JMS standard - allows only a single consuming client thread.	10.3	Non-shared durable Connection Factory	No
Serial	Enforces serial consumption across 1 .. n subscribers	10.3	Serial durable Connection Factory	Yes

Software AG Universal Messaging License Features

Software AG Universal Messaging has three license types. Choose a license type based on your requirement. You can switch to a more advanced license type at any time without reinstalling or reconfiguring. The following table lists the features of the Software AG Universal Messaging license types. Contact Software AG to obtain the correct license.

Universal Messaging for webMethods Integration License (NUMWI) Features (Equivalent to Broker)	**Universal Messaging for webMethods Integration License with Active/Active Add-On (NUMWS) Features**	**Software AG Universal Messaging Fully-featured License (NUMWF) Features**
■ JMS connectivity ■ Active/Passive clustering ■ Java, C++, C# APIs ■ Socket-based transport (with or without SSL) ■ Server-to-server forwarding	■ JMS connectivity ■ Active/Passive clustering ■ Active/Active clustering ■ Java, C++, C# APIs ■ Socket-based transport (with or without SSL) ■ Server-to-server forwarding	■ JMS connectivity ■ Active/Passive clustering ■ Active/Active clustering ■ Web and mobile application support ■ High performance in-memory communication options ■ Java, C++, C# APIs ■ Socket-based transport (with or without SSL) ■ Server-to-server forwarding ■ HTTP(S) transport ■ Multicast transport

Triggers

Triggers that are migrated to Universal Messaging will function the same as they were in Broker, but will use Integration Server client-side filtering. Using Integration Server client-side filtering may impact the performance of your messaging environment. You can switch to server-side filtering provided by the Universal Messaging server-side filtering engine to improve performance.

To switch to server-side filtering, follow these steps:

- **Universal Messaging 10.1 and later**
  - Use the pub.utils.messaging:migrateDocTypesTriggersToUM service in Designer. For more information, see "Document Type and Trigger Migration" on page 32.

Security

- **Basic authentication**

If you are using basic authentication in Broker to secure Broker Servers, you can configure basic authentication in Universal Messaging 10.1 and later. User name and password credentials can be passed from Universal Messaging clients to the Universal Messaging server.

Basic authentication credentials can be used for JMS connections from Integration Server 9.6 and later. webMethods Messaging (native publish-subscribe) connections to Universal Messaging support basic authentication credentials in Integration Server 9.8 and later.

- **SSL encryption**

If you are using SSL to encrypt connections to Broker Servers, you can configure Universal Messaging interfaces to use SSL. Defining an SSL enabled interface in Universal Messaging ensures that clients wishing to connect to a server can do so only after presenting the correct SSL credentials and authenticating with the server. You can define an SSL enabled interface using Universal Messaging Enterprise Manager.

For information about SSL encryption and basic authentication in Universal Messaging, see *Universal Messaging Concepts Guide*. For instructions to configure SSL and basic authentication, see the Universal Messaging Webhelp on the Software AG documentation website at http://documentation.softwareag.com.

Zones and Joins

Universal Messaging zones are similar to Broker territories. You can join Universal Messaging servers or clusters to form a gateway that is similar in functionality to a Broker gateway. For information about migrating from Broker territories and gateways to Universal Messaging zones and joins, see " Migrating to Zones and Joins" on page 34.

Clustering

Universal Messaging supports the following clustering approaches:

- **Active/active cluster**
  - Universal Messaging cluster with an odd number of servers
  - Universal Messaging cluster with an even number of servers across two sites
- **Active/Passive cluster**
  - Universal Messaging cluster with shared storage

Universal Messaging clients can connect to a cluster using a defined failover sequence or using random distribution across the cluster, depending on the URL format used to connect to the cluster.

Note:webMethods Messaging connections for Integration Server 9.7 and earlier support only the defined failover sequence.

For more information about Universal Messaging clustering, see *Universal Messaging Administration Guide*.

2 Migrating from Broker to Universal Messaging

Pre-migration Considerations

Identify and install webMethods products that you intend to use with Universal Messaging. For example, Integration Server, Deployer, and Optimize.

Note:

- You must install all the latest fixes before starting the migration.
- For information on how to install a new product and start migration, refer the *Install New Products and Begin Migration* section in *Upgrading Software AG Products* document.

Broker Considerations

The following table lists the differences between Broker and Universal Messaging that you must consider before migration.

Broker	Universal Messaging	Recommendation
webMethods Messaging on an Integration Server supports connecting to only one Broker instance.	webMethods Messaging on an Integration Server supports connecting to multiple Universal Messaging server instances. This increases the scalability of the messaging platform.	During migration, individual Broker instances in a Broker Server can be migrated to a single or multiple Universal Messaging realms.
Broker uses My webMethods Server for administration.	Universal Messaging uses Enterprise Manager for administration. Software AG Command Central can also be used to administer and deploy Universal Messaging.	The functionality differs for Universal Messaging. Refer "Universal Messaging Administration" on page 20 section for more information.
Broker supports Territories and Gateways.	Universal Messaging supports equivalent functionality through Zone and Static Joins.	For more information about how Universal Messaging uses Zone and Static Joins, refer to "Universal Messaging Zone and Static Joins" on page 20.
Broker clients native messages are encoded by Integration Server using IData.	Universal Messaging uses the Google Protocol Buffer (protobuf) for the native messages.	Some of the Broker field names do not have the exact equivalent syntax which is available in Google Protocol Buffer field name. For information about how Universal Messaging uses Google Protocol Buffer (protobuf), see "Google Protocol Buffer Specification" on page 21.

Broker	Universal Messaging	Recommendation
Broker supports filtering with standard comparison, arithmetic operators, bitwise operators, string operators, and standard operator precedence. Also, Broker filters support regular expressions, hints, and some in-built functions.	Universal Messaging also supports filters, along with its extensions.	Some Broker filter extensions does not have equivalent functionality in Universal Messaging. For information refer, "Filtering" on page 21. Refer "Event Filtering" section in *Universal Messaging Concepts Guide* for more information on filtering.
Broker supports XA transactions (2 phase commit).	Universal Messaging does not support XA transactions.	In most of the cases, you can workaround the XA limitation by analyzing your transaction boundaries and by using the last resource commit optimization. For more information on XA transaction, refer *webMethods Integration Server Administrator's Guide*.
Broker supports message priority.	Universal Messaging provides limited support for message priority. It prioritizes messages on the client, which means that the messages may not be prioritized as you expect.	However, in Universal Messaging you can use the alternate approaches. For instance, you can use multiple topics for specific priority and preferentially consuming the messages from the higher-priority topics.
Broker supports individual subscription pause when the corresponding Integration Server trigger is paused.	Universal Messaging does not completely support pausing individual subscriptions.	Pausing a Integration Server trigger with Universal Messaging can cause duplication of messages. For information on how the Universal Messaging works with limited support on pausing individual subscriptions, refer "Pause Integration Server Triggers " on page 23.
Broker supports Document Logging for auditing and resubmitting documents.	Universal Messaging does not support Document Logging.	For Universal Messaging, you can use Integration Server Service Flow auditing functionality to achieve the same.

Broker	Universal Messaging	Recommendation
		For more information see, "About Service Auditing" section in *Software AG Designer Online Help*.
Broker supports Dead Letter Queue.	Universal Messaging also supports the Dead Letter Queue. However, the functionality differs.	For more information on Dead Letter Queue, see "Dead Letter Queue" on page 24.
Broker supports horizontal scalability using Clustering with commonly used Round Robin policy.	Universal Messaging supports horizontal scalability which is used in Round Robin JMS Cluster policy for Broker.	For information on how differently Universal Messaging uses horizontal scalability see,"Horizontal Scalability" on page 24.
Broker supports high availability using the Mutlisend cluster policy.	Universal Messaging supports Active/Active cluster to provide high availability.	For more information, see "Clustering" section in *Universal Messaging Administration Guide* documentation.
Broker supports proprietary Java, C#, and C API libraries.	Universal Messaging also supports proprietary Java, C#, and C API libraries. But, the exact interface definition differs.	Ensure to verify all your custom client applications that make use of Broker Java, C#, C APIs and plan the migration.

Universal Messaging Administration

Universal Messaging uses Enterprise Manager as its administration and monitoring tool. Enterprise Manager can run remotely and manages multiple Universal Messaging servers. Enterprise Manager uses Universal Messaging Administration APIs, similarly for Broker, My webMethods User Interface uses Broker Administration APIs. If you have custom Broker administration clients, you need to rewrite that client using Universal Messaging Administration APIs.

In addition, Universal Messaging also supports JMX API.

Universal Messaging Zone and Static Joins

Brokers can be linked to form units known as territories. Territory gateways are links that you establish between territories. Now, the territories and gateways are replaced with Universal Messaging Zones and Universal Messaging Joins respectively. Therefore, you can establish a similar messaging topology with Universal Messaging or you can review and simplify your messaging landscape when you migrate.

Universal Messaging administration and monitoring support for a large number of topics and queues are limited, so you must plan carefully and consider changing your topology. Universal Messaging Zones and Static Joins are implemented differently in Universal Messaging. During the migration you must ensure to understand and accommodate to these changes. For example,

to set up request-reply across Universal Messaging servers, you need to perform additional steps for creating required topics and queues.

Google Protocol Buffer Specification

Google Protocol Buffer Specification provides a simple and performant serialized format for transporting data across the network. The Google Protocol Buffer specification contains support for a limited character set. Hence, ensure that the names and fields in your document comply with the protobuf spec https://developers.google.com/protocol-buffers/docs/reference/proto3-spec.

Further issues related to using the protocol buffer encoding type are available in the section "Using Protocol Buffers as the Encoding Type" of *Software AG Designer Online Help*.

The following are the commonly encountered issues:

- *Spaces*- Spaces appear in your document or field names. Remove the spaces if it is not required.
- @- The @ character may appear in your document or field names which can cause warning messages. Remove this character if it is not required.
- *Colons(:)*- Integration Server uses colons for XML namespace identification. If you do not use XML, change them to protobuf acceptable characters. Or, if you are using XML, then it would assess the impact on performance of performing the client side filtering rather than server side filtering.

Filtering

The migration utility pub.utils.messaging:migrateDocTypesTriggersToUM attempts to migrate the filter to a Universal Messaging (provider) filter. However, this process is not always achieved. Sometimes, the filter definition from Integration Server or Broker may not meet the SQL/92 filter standards supported by Universal Messaging. Also, the filter definition from Integration Server or Broker uses some extensions that are not supported by Universal Messaging. This can be a considerable task if there are large number of clients and flow services in the affected document types.

If you find any errors or warning while running the pub.utils.messaging:migrateDocTypesTriggersToUM utility for the field names and identifiers, rename the fields to valid identifiers for Google Protocol Buffers and perform the migration for the affected documents and other objects.

In conjunction with Designer, Integration Server also supports the ability to search for a particular variable and identify all references to that variable in other assets, such as flow services, document types, specifications, and triggers. You can selectively or globally replace a particular variable name with another variable name. This can considerably reduce the effort required to rename the fields. Perform a back-up for all the affected packages before you refactor.

You can still continue to use the same fields without renaming, but these fields are migrated with filters as Integration Server (subscriber) filters. However, large number of triggers having different filter criteria can potentially impact Integration Server processing. Make sure these filters are working and are verified with a load similar to that expected in the production.

Important:

- If migration utility pub.utils.messaging:migrateDocTypesTriggersToUM fails to migrate Broker's server side filters to Universal Messaging server side filtering, then it is mapped to client side filtering. This may impact the performance.
- Broker filters also support regular expressions or hints such as IncludeDeliver, LocalOnly, DeadLetterOnly, and some functions like substring, toDouble, toUpperCase, etc. These may not have an equivalent mapping in Universal Messaging. In your migration, you must factor in the impact of such in-frequently used Broker filter extensions.

Comparison of filter operators in Broker and Universal Messaging

Refer to the below tables to find the full list of compatible and non-compatible filter operators.

The following table shows the comparison of arithmetic filter operators:

Operator name	Broker filter	Universal Messaging filter
Addition	`+`	`+`
Subtraction	`-`	`-`
Multiplication	`*`	`*`
Division	`/`	`/`
Modulo	`%`	Unsupported
Unary minus	`-`	Unsupported

The following table shows the comparison of relational/comparison filter operators:

Operator name	Broker filter	Universal Messaging filter
Equal to	`=,==`	`=`
Not equal to	`!=`	`<>`
Greater than	`>`	`>`
Less than	`<`	`<`
Greater than or equal to	`>=`	Unsupported
Less than or equal to	`<=`	Unsupported

The following table shows the comparison of logical filter operators:

Operator name	Broker filter	Universal Messaging filter
Logical negation	`!, not`	`NOT`

Operator name	Broker filter	Universal Messaging filter
Logical AND	&&, and	AND
Logical OR	||, or	OR

The following table shows the comparison of bitwise filter operators:

Operator name	Broker filter	Universal Messaging filter
Bitwise NOT	~	Unsupported
Bitwise AND	&	Unsupported
Bitwise OR	|	Unsupported
Bitwise XOR	^	Unsupported
Bitwise left shift	<<	Unsupported
Bitwise right shift	>>	Unsupported

The following table shows the comparison of string filter operators:

Operator name	Broker filter	Universal Messaging filter
String concatenation	+	Unsupported
Equal to	=	Unsupported
Equal to	==	Unsupported
Not equal to	!=	Unsupported
Greater than	>	Unsupported
Greater than or equal to	>=	Unsupported
Less than	<	Unsupported
Less than or equal to	<=	Unsupported

For more information on filters in Broker, see *webMethods Broker Client Java API Programmer's Guide*. For more information on filters in Universal Messaging, see *Universal Messaging Administration Guide*.

Pause Integration Server Triggers

For Universal Messaging, pausing Integration Server triggers frequently can result in duplicate messages at triggers. During trigger pause, Integration Server can cease subscribing and then re-subscribe. This can disrupt the acknowledgement flow to Universal Messaging which can result

to a message re-delivery. Ensure that your applications are aware of the messages and event IDs to detect duplicate messages.

Dead Letter Queue

For Dead Letter, processing the events in Broker without any subscription or failed filter condition results in Dead letter queue.

In Universal Messaging, the events move to dead event store when they are removed by Universal Messaging realm, for cases such as, channel capacity, channel TTL, event TTL.

Horizontal Scalability

Horizontal Scalability in Universal Messaging allows the clients to seamlessly publish and consume events from multiple independent realms and clusters using a single connection. It is available for both the Universal Messaging native API for Java and the Universal Messaging API for JMS. It is enabled by using the horizontal scalability URL syntax. For information, refer *Universal Messaging Administration Guide* documentation.

Other Product Considerations

Process Engine Trigger Retry Setting

When the Subscription Trigger is configured to retry until *Success* or has an infinite retry loop, then a document that fails to correlate into a running instance is replayed forever. Integration Server trigger may appear stuck. Universal Messaging does not show the document that is pending acknowledgement, so that the document appears lost. Upon restarting Integration Server, the document is processed again and the cycle may repeat.

To resolve this issue, in Process Engine you can configure the server configuration property watt.prt.suppressCorrelationRetry to *true*. This suppresses the correlation for a retry. Instead of a transient retry, a regular exception is thrown so there is visibility to the correlation error.

Note:
The watt.prt.suppressCorrelationRetry property changes the default behaviour.

JMS and webMethods Messaging

Universal Messaging can be used for:

- JMS Messaging
- webMethods Messaging
- JMS Messaging and webMethods Messaging

JMS Messaging

Universal Messaging can be configured for JMS messaging. If you are using Broker for JMS messaging, you can migrate to Universal Messaging. Integration Server uses pub.jms:* services and JMS triggers for JMS messaging. When Universal Messaging is configured as a JMS provider, Integration Server uses JNDI lookup for connection factories and destinations. For migration information and instructions, see "JNDI and JMS Migration" on page 25.

webMethods Messaging

webMethods Messaging, also known as native publish-subscribe, is supported in Universal Messaging. If you are using Broker for webMethods Messaging, you can migrate to Universal Messaging. For information and migration steps, see " webMethods Messaging Migration" on page 31.

JNDI and JMS Migration

Use the migration scripts to migrate the JNDI and JMS assets (listed in "JNDI Migration Capabilities" on page 25 and "JMS Migration Capabilities" on page 26). Install the latest Universal Messaging fixes to ensure that you have the latest migration scripts. The migration scripts are installed in the following directory: *`Universal Messaging_directory`*`/tools/migrate`.

The migration scripts write the following detailed migration information to the log files:

- Names of the assets successfully migrated
- Names of the assets not migrated and the reason why an asset was not migrated
- Warning messages when there is a mismatch between the assets or properties of the assets in the source Broker Server and the target Universal Messaging server
- Migration error messages
- Migration information

The migration scripts does not migrate the messages and configurations present in the source Broker.

JNDI Migration Capabilities

To migrate the JNDI assets from Broker to Universal Messaging, use the brokerjndimigration.{bat/sh} migration script. The script does the following:

- Migrates the following JNDI assets from Broker used as a JNDI provider to a target Universal Messaging server used as a JNDI provider:
  - Connection factories
  - Topics

  - Queues
- Writes the following migration information to the brokerjndimigrationout.txt log file in the same directory as the migration scripts:
  - List of connection factories, topics, and queues of the source JNDI provider that were successfully migrated to the target JNDI provider.
  - Warning messages if there is a mismatch between the properties of the source JNDI assets and the target JNDI assets.
- Writes the error messages to the brokerjndimigrationerr.txt file in the same directory as the migration scripts.

For more information about running the JNDI migration script, see "Running the JNDI Migration Script" on page 27.

JMS Migration Capabilities

To migrate the JMS assets from Broker to Universal Messaging, use the brokerjmsmigration.{bat/sh} migration script. The script does the following:

- Migrates all the following JMS assets from a source Broker to the target Universal Messaging server:
  - Topics
  - Queues

    Note:
    JNDI destinations for topics and queues are auto-generated for the migrated JMS assets in the target Universal Messaging server.
  - Client groups

    The Broker client groups are migrated as security groups to Universal Messaging. For topics with durable subscribers, the durable subscribers will be re-created in Universal Messaging when the client connects.
- Writes the following migration information to the brokerjmsmigrationout.txt log file in the same directory as the migration scripts:
  - List of JMS topics, queues, client groups, and the corresponding JNDI Destinations of the source Broker that were successfully migrated to the target Universal Messaging server.
  - Warning messages if there is a mismatch between the properties of the JMS assets in the source Broker and the target Universal Messaging server.
- Writes the error messages to the brokerjmsmigrationerr.txt file in the same directory where the migration scripts are placed.

For more information about running the JMS migration script, see "Running the JMS Migration Script" on page 29.

Before you Run the Migration Scripts

- Disable security on the source Broker and the target Universal Messaging server.
- Ensure that the source Broker Server and the newly installed target Universal Messaging server are running.

> **Note:**
> Running the migration scripts will not affect the messaging traffic in the source Broker.

Running the Migration Scripts

After you complete the tasks described in "Before you Run the Migration Scripts" on page 27, perform these tasks to migrate the JNDI and JMS assets from Broker to Software AG Universal Messaging:

1. Ensure that both the source Broker and target Universal Messaging servers are started and running.

2. Run the JNDI migration script to migrate the JNDI assets from Broker to the target JNDI provider. See "Running the JNDI Migration Script" on page 27.

3. Check for error messages in the JNDI migration script error log file, if there are errors, make the required corrections, and re-run the JNDI migration script.

4. Run the JMS migration script to migrate the JMS assets from a source Broker to the target Universal Messaging server. See "Running the JMS Migration Script" on page 29.

5. Check the error messages in the JMS migration script error log file, if there are errors, make the required corrections, and re-run the JMS migration script.

6. Check the JNDI migration and the JMS migration messages in the log files to understand which assets were successfully migrated. Based on the warning messages, update the migrated assets in the Universal Messaging servers. See "JNDI and JMS Migration Warnings" on page 57.

7. Perform the post-migration steps. See "Performing Post-Migration Configuration" on page 39.

Running the JNDI Migration Script

Do one of the following to complete JNDI migration:

- Run the brokerjndimigration Script
- Run the brokerjndimigration Script with Command-Line Arguments
- Use the brokerjndimigration Script with a Properties File

> **Note:**

To display command-line help, in the command prompt, type **brokerjndimigration.bat ?**

Run the brokerjndimigration Script

1. On the machine that hosts the target Universal Messaging server, go to the `Universal Messaging_directory/tools/migrate` directory and run the brokerjndimigration script.

2. The script prompts: Enter Broker JNDI Provider URL

   Enter the URL of the source Broker from which you want to migrate the JNDI assets to the target JNDI provider.

3. The script prompts: Enter Target JNDI Provider URL

   Enter the URL of the target Universal Messaging server. Specify the fully qualified name or IP address of the host machine.

   The JNDI migration script runs and writes information to the brokerjndimigrationout.txt log file and the brokerjndimigrationerr.txt error log file.

4. The script prompts: Do you want to continue. Enter yes/no

   Type **yes** to continue migration. Type **no** to re-enter the target JNDI provider URL.

   Note:
   If you use **localhost** or *localhost_ip_address* as JNDI provider URL, only local clients can connect. If you are using a remote Broker, enter a valid JNDI provider URL.

Example

```
C:\SoftwareAG\UniversalMessaging\tools\migrate>brokerjndimigration.bat
Enter Broker JNDI Provider URL
wmjmsnaming://Broker #1@localhost:6849
Enter Target JNDI Provider URL
nsp://localhost:9000
Recommended URL format: nsp://umserverhost:9000 or nsp://umserverip:9000
If you use localhost or localhost_ip_address, only the local clients can
connect.
Do you want to continue with the localhost/localhost_ip_address? Enter yes/no
yes
```

Run the brokerjndimigration Script with Command-Line Arguments

Use brokerjndimigration command and provide command line arguments.

1. Open a Command Prompt window and navigate to *Universal Messaging_directory* /tools/migrate directory.

2. Type **brokerjndimigration.bat -brokerjndiurl "***broker jndi provider url***" -targetjndiurl "***target jndi provider url***"**

Example

```
C:\SoftwareAG\UniversalMessaging\tools\migrate>brokerjndimigration.bat
-brokerjndiurl "wmjmsnaming://Broker #1@localhost:6849"
-targetjndiurl "nsp://umserverhost:9000"
```

Use the brokerjndimigration Script with a Properties File

Use a properties file to pass migration parameters to brokerjndimigration.

1. Open a Command Prompt window and navigate to *Universal Messaging_directory* /tools/migrate directory.

2. Type **brokerjndimigration.bat -f** "*path to the properties file*"

   The properties files should contain migration parameters in the following format:

   ```
   brokerjndiurl="broker jndi provider url"
   ```

   ```
   targetjndiurl="target jndi provider url"
   ```

   ```
   sourceinitialcf="source initial connection factory"
   ```

   ```
   targetinitialcf="target initial connection factory"
   ```

   Note:
   Provide `sourceinitialcf` and `targetinitialcf` when your source JNDI is not a Broker and your target JNDI is not a Universal Messaging server.

Example

Parameters in a properties file

```
brokerjndiurl="wmjmsnaming://Broker #1@localhost:6849"
targetjndiurl="nsp://umserverhost:9000"
```

Optional parameters

```
sourceinitialcf="com.sun.jndi.ldap.LdapCtxFactory"
targetinitialcf="com.sun.jndi.ldap.LdapCtxFactory"
```

Command line

```
brokerjndimigration.bat -f "C:\MigrationProperties\JNDI\input.properties"
```

Running the JMS Migration Script

Do one of the following to complete JMS migration:

- Run the brokerjmsmigration Script
- Run the brokerjmsmigration Script with Command-Line Arguments

- Run the brokerjmsmigration Script with a Properties File

Note:
To display command-line help, in the command prompt, type **brokerjmsmigration.bat ?**

Run the brokerjmsmigration Script

1. On the machine that hosts the target Universal Messaging server, go to the *Universal Messaging_directory* /tools/migrate directory and run the brokerjmsmigration.{bat/sh} script.

2. The script prompts: Enter Broker Host and Port

   Enter the host name and the port number of the source Broker from which you want to migrate the JMS assets.

3. The script prompts: Enter Broker Name

   Enter the case-sensitive name of the source Broker.

4. The script prompts: Enter Universal Messaging [UM] URL

   Enter the URL of the target Universal Messaging server.

5. The script prompts: Do you want to auto-generate JNDI Destinations on target Universal Messaging [UM]?

   Enter **yes** or **no**

   The JMS migration script runs and writes information to the brokerjmsmigrationout.txt log file and the brokerjmsmigrationerr.txt error log file.

Example

```
C:\SoftwareAG\nirvana\tools\migrate> brokerjmsmigration.bat
Enter Broker Host and Port
localhost:6849
Enter Broker Name
Broker #1
Enter Universal Messaging[UM] URL
nsp://localhost:9000
Do you want to auto-generate JNDI Destinations on target Universal Messaging[UM]?
yes
```

Run the brokerjmsmigration Script with Command-Line Arguments

Use brokerjmsmigration.{bat/sh} command and provide command line arguments.

1. Open a Command Prompt window and navigate to *Universal Messaging_directory* /tools/migrate directory.

2. Type **brokerjmsmigration.bat -host** "*broker host:port*" **-broker** "*broker name*" **-targeturl** "*target universal messaging server address*" **-autogeneratejndi** *yes/no*

> **Note:**
> Auto-generation of JNDI destinations for the migrated topics and queues is enabled by default.

Example

```
C:\SoftwareAG\UniversalMessaging\tools\migrate>brokerjmsmigration.bat
-host "localhost:6849"
-broker "Broker #1"
-targeturl "nsp://localhost:9000"
```

Run the brokerjmsmigration Script with a Properties File

Use a properties file to pass migration parameters to brokerjmsmigration.{bat/sh}.

1. Open a Command Prompt window and navigate to *Universal Messaging_directory* /tools/migrate directory.

2. Type **brokerjmsmigration.bat -f** "*path to the properties file*"

   The properties files should contain migration parameters in the following format:

   ```
   host="brokerhostname:port"
   ```

   ```
   broker="broker name"
   ```

   ```
   targeturl="target universal messaging url"
   ```

   ```
   autogeneratejndi="yes/no"
   ```

Example

Parameters in a properties file

```
host="localhost:8849"
broker="jmsassetbroker"
targeturl="nsp://localhost:9000"
autogeneratejndi=yes
```

Command line

```
brokerjmsmigration.bat -f "C:\MigrationProperties\JMS\input.properties"
```

webMethods Messaging Migration

You can choose from one of the following approaches when migrating from Broker to Universal Messaging and using Universal Messaging for webMethods Messaging. You can either completely

migrate to Universal Messaging and make it the only messaging provider or migrate to Universal Messaging in phases with Broker being the default messaging provider.

Approach One

Make Universal Messaging your only messaging provider. To use Universal Messaging as the only messaging provider:

1. Configure Universal Messaging connection in Integration Server and set it as default.

2. Sync the document types using Software AG Designer.

3. Run the pub.utils.messaging:migrateDocTypesTriggersToUM service to optimize Integration Server document types and triggers for use with Universal Messaging (Recommended).

Approach Two

Migrate to Universal Messaging in phases. To use Universal Messaging alongside Broker and set Broker as your default messaging provider:

1. Configure Universal Messaging connection alongside the default Broker connection. Ensure that you have enabled both Universal Messaging and Broker in the Integration Server messaging settings and set Broker as the default messaging provider.

2. Switch document types selectively by changing messaging provider alias from Broker to Universal Messaging using Software AG Designer.

3. Run the pub.utils.messaging:migrateDocTypesTriggersToUM service to optimize Integration Server document types and triggers for use with Universal Messaging (Recommended).

Document Type and Trigger Migration

The pub.utils.messaging:migrateDocTypesTriggersToUM service can be found in the WmPublic folder in Designer. Run this service to migrate publishable document types and the subscribing triggers to use Universal Messaging as the messaging provider.

Note:
This service works only with Universal Messaging 9.8 and later.

The pub.utils.messaging:migrateDocTypesTriggersToUM service can be used to perform one or more of the following:

- Change the messaging connection alias assigned to the publishable document types to Universal Messaging.

  Note:

This step is required only if you choose to have Broker as your default messaging provider and migrate to Universal Messaging in phases.

- Set the encoding type of the publishable document type to protocol buffers.
- Synchronize the updated publishable document types with Universal Messaging.
- Convert the filters used by the webMethods messaging triggers that subscribe to the publishable document types. Specifically, the service migrates the filter expressions that can be evaluated by the Universal Messaging server from the **Filter** field for a trigger condition to the **Provider Filter (UM)** field.

Before running the service:

- Ensure that Integration Server is connected to Broker.
- Ensure that the document types are in sync with their associated Broker document types in the target Integration Server.
- Determine the list of publishable document types to be migrated. Migrate all document types subscribed by a group of webMethods messaging triggers at one time.
- Ensure that if a trigger subscribes to more than one publishable document type, all the publishable document types use the same messaging connection alias. Otherwise,Integration Server considers the trigger to be invalid.
- Ensure that the webMethods messaging triggers are not locked for edit by another user when running this service.
- Ensure that you have Write access to the publishable document types and webMethods messaging triggers.
- Decide if you want to use *packageNames* or *documentTypeNames* input parameters. If you specify a value for both parameters, the service will fail with an exception.
- Run the service in report only mode by setting the *reportOnly* input parameter to true. Use the service output to find potential problems in the migration.

Before using this service to migrate document types and triggers, see the information and usage notes in *webMethods Integration Server Built-In Services Reference*.

Territories and Gateways Migration

If you are using territories and gateways in Broker, you can migrate your infrastructure to use similar features by configuring zones and joins in Universal Messaging. Run the Broker gateway migration utility to automatically migrate gateway configuration from a source Broker to a Universal Messaging server instance.

Note:
The Broker gateway migration utility is available as part of the installation only in Universal Messaging 10.0 and later.

Migrating to Zones and Joins

To migrate your Broker territories and gateways to zones and joins in Universal Messaging:

1. Create zones and define the set of Universal Messaging servers or clusters that form a zone.

   You can map and migrate your Broker territories by creating an equivalent number of Universal Messaging zones.

   Note:
   Document types, channels, and topics are not automatically replicated in a zone. When using Universal Messaging as a JMS provider, you create or deploy channels and topics on all zone members where clients can publish or subscribe. For webMethods Messaging, synchronize publishable document types from at least one Integration Server connected to each zone member.

2. Link two Universal Messaging servers or clusters by choosing and adding one Universal Messaging server or cluster to the another server or cluster, this will create a link between the two Universal Messaging server or cluster similar to a Broker gateway.

3. Create channel joins between the two Universal Messaging servers or clusters using the Universal Messaging Enterprise Manager.

   Setting up a channel join between two servers will create a one-way join. To allow messages to be exchanged in both directions, create channel joins in both directions.

To migrate gateway configuration to your Universal Messaging server instances, run the gateway configuration migration script, see Gateway Migration. For more information about zones and joins, see *Universal Messaging Administration Guide*.

Gateway Migration

You can migrate your Broker gateway configuration from your source Brokers to Universal Messaging server instances using the gateway configuration migration script. The script migrates the following:

- Shared document types in the source Broker to corresponding remote joins in Universal Messaging
- Broker cluster gateway configuration to the target Universal Messaging
- Broker filters to corresponding Universal Messaging filters

Note:
If the join is created from a channel to a queue, the join is an archival join where the events will not be checked for duplication.

Running the Gateway Configuration Migration Script

The Broker Gateway Migration Utility supports automatic migration of certain operators that you input in the shared document type filters. But there are some operators that you need to manually modify to have an error-free migration process. For more information on filter operator compatibility, see "Comparison of filter operators in Broker and Universal Messaging " on page 22.

Important:
The filter operator conversions that are listed occur regardless of whether the operator resides within the specified values or not.

Before running the gateway configuration migration script, ensure that you run the JNDI and JMS migration scripts. For more information on running the JNDI and JMS migration scripts, see "JNDI and JMS Migration" on page 25.

Do one of the following to complete gateway configuration migration:

- Run the brokergatewaymigration Script
- Run the brokergatewaymigration Script with Command-Line Arguments
- Use the brokergatewaymigration Script with a Properties File

Note:
To display command-line help, in the command prompt, type **brokergatewaymigration.bat**

Run the brokergatewaymigration Script

1. Ensure that the operators present in your shared document type filters are intercompatible with Broker and Universal Messaging. The following table lists the operators that are migrated automatically by the script. You need to manually replace the operators that are not mentioned in the table.

Operator	Broker filter	Universal Messaging filter
Equal to	`=,==`	`=`
Not equal to	`!=`	`<>`
Logical negation	`!, not`	`NOT`
Logical AND	`&&, and`	`AND`
Logical OR	`||, or`	`OR`

2. On the machine that hosts the target Universal Messaging server, go to the *`Universal Messaging_directory`*`/tools/migrate` directory, and run the brokergatewaymigration script.

3. The script prompts: **Enter Broker Host and Port**

   Enter the Broker host name and port in the format: *broker host:broker port* . For example, brokerhost:6849.

4. The script prompts: **Enter the source Broker Name**

   Enter the name of the source Broker in the format:*broker name*. For example, broker #1.

5. The script prompts: **Enter Source Territory/Cluster**

   Enter the name of the source territory in the format: *sourceterritory*.

6. The script prompts: **Enter Remote Territory/Cluster**

   Enter the name of the remote territory in the format: *remoteterritory*.

7. The script prompts: **Enter the Universal Messaging URL that contains the source channel/topic for the join**

   Enter the Universal Messaging URL that contains the channel or topic from which you want to create a join. For example, nsp://umserverhost:9000.

8. The scripts prompts: **Enter the Universal Messaging URL that contains the target channel/topic for the join**

   Enter the Universal Messaging URL that contains the channel or topic to which you want to create a join. For example, nsp://umserverhost:9001.

The Broker gateway migration script will run and display the results of the migration.

Example

```
C:\SoftwareAG\UniversalMessaging\tools\migrate>brokergatewaymigration.bat
Enter Broker Host and Port
brokerhost:6849
Enter source Broker Name [For example, Broker #1]
broker1
Enter Source Territory/Cluster
territory1
Enter Remote Territory/Cluster
terriotry2

Enter the Universal Messaging URL that contains the source
channel/topic for the join [For example, nsp://umserverhost:9000]
nsp://localhost:1001

Enter the Universal Messaging URL that contains the target
channel/topic for the join
nsp://localhost:1002
```

Run the brokergatewaymigration Script with Command-Line Arguments

Use brokergatewaymigration command and provide command line arguments.

1. Open a Command Prompt window and navigate to *Universal Messaging_directory* /tools/migrate directory.

2. Type **brokergatewaymigration.bat-host** *"brokerhost:port"* **-broker** *"broker name"* **-source** *"Source territory or cluster"* **-remote** *"Remote territory or cluster"* **-targeturl1** *"Universal Messaging URL that contains the source channel/topic for the join"* **-targeturl2** *"Universal Messaging URL that contains the target channel/topic for the join"*

Example

```
C:\SoftwareAG\UniversalMessaging\tools\migrate>brokergatewaymigration.bat
  -host "brokerhost:6849"
  -broker "Broker #1
  -source "territorysource"
  -remote "territoryremote"
  -targeturl1 "nsp://localhost:1001"
  -targeturl2 "nsp://localhost:1002"
```

Use the brokergatewaymigration Script with a Properties File

Use a properties file to pass migration parameters to brokergatewaymigration.

1. Open a Command Prompt window and navigate to *Universal Messaging_directory* /tools/migrate directory.

2. Type **brokergatewaymigration.bat -f** "*path to the properties file*"

   The properties files should contain migration parameters in the following format:

   ```
   host="Broker host and port"
   ```

   ```
   broker="Broker name"
   ```

   ```
   source="Source territory or cluster"
   ```

   ```
   remote="Remote territory or cluster"
   ```

   ```
   targeturl1="Universal Messaging URL that contains the source
   channel/topic for the join"
   ```

   ```
   targeturl2="Universal Messaging URL that contains the target
   channel/topic for the join"
   ```

Example

Parameters in a properties file

```
host="brokerhost:6849"
broker="Broker #1"
source="territorysource"
remote="territoryremote"
targeturl1="nsp://localhost:1001"
targeturl2="nsp://localhost:1002"
```

Command line

```
brokergatewaymigration.bat -f "C:\MigrationProperties\gateway\input.properties"
```

3 Performing Post-Migration Configuration

Overview

This section describes the post-migration steps you need to perform to use Software AG Universal Messaging instead of Broker.

webMethods product	Post-Migration Configuration Task
Universal Messaging	To enable Universal Messaging for JMS messaging, configure: ■ JNDI namespace ■ JMS destinations ■ JMS clients For more information, see " Universal Messaging Configuration" on page 41.
Integration Server	To enable the connection between Integration Server and Universal Messaging for JMS messaging, configure: ■ DEFAULT_IS_JNDI_PROVIDER alias or new JNDI provider alias ■ DEFAULT_IS_JMS_CONNECTION alias or new JMS connection alias To enable webMethods Messaging, configure: ■ Document Types ■ Triggers For more information, see " Integration Server Configuration" on page 41 and " webMethods Messaging" on page 25.
Optimize	To enable Universal Messaging server discovery, data communication, asset discovery, and server monitoring, configure: ■ JMS Server URL property ■ Universal Messaging asset discovery For more information, see " Optimize Configuration" on page 46.
Process Engine	To enable Universal Messaging as the JMS server for publishing the audit messages to Optimize, configure the `com.softwareag.eda.nerv.default.jms.provider` property. For more information, see " Process Engine Configuration" on page 49.
Command Central	To enable centralized Universal Messaging administration and monitoring, configure Universal Messaging server installations and instances in Command Central. For more information, see *Software AG Command Central Help*.

webMethods product	Post-Migration Configuration Task
Designer	To enable design-time Universal Messaging destination management, JMS trigger creation, and JMS message mapping with the process models, configure the JMS triggers and process models to change the JMS connection alias. For more information, see *webMethods Service Development Help* and *webMethods Integration Server Administrator's Guide*.
Deployer	To enable Repository-based deployment, configure Deployer connection to the Universal Messaging server. For more information, see *webMethods Deployer User's Guide*.

Universal Messaging Configuration

Use Universal Messaging Enterprise Manager to:

- Update the migrated assets based on the migration messages. For descriptions of the warnings logged in the JNDI and JMS migration log files, see "JNDI and JMS Migration Warnings" on page 57.

  Important:
  Make sure you select the JNDI Lookup option for the JMS connection alias in Integration Server.

- Select the **Connection Factory (Shared Durable)** option when you create the connection factory, if you want to use durable subscribers.
- Set the MaxBufferSize configuration property to be large enough for the largest message you want Universal Messaging to transport. The MaxBufferSize property is set to 1 MB by default. An Integration Server client will be disconnected fromUniversal Messaging server when a message exceeding the MaxBufferSize is transmitted.
- Increase the client queue window size if you want a concurrent Integration Server trigger to process more than 10 documents at a time. The default Universal Messaging window size is 10 documents. For example, if a concurrent trigger has max execution threads set to 30, set the client queue window size to 30.

For more information about using Universal Messaging Enterprise Manager, see the Universal Messaging Webhelp on the Software AG Documentation website at http://documentation.softwareag.com.

Integration Server Configuration

You have to configure JNDI and JMS if you are using Universal Messaging as a JMS provider. When using webMethods Messaging, configure Universal Messaging connection in Integration Server.

To configure Integration Server to use Universal Messaging as the JMS provider in place of Broker, complete the following tasks:

1. Configure a JNDI provider alias to:

   - Instruct Integration Server where to look up administered objects when it needs to create a connection to the Universal Messaging server used as a JMS provider.
   - Specify Universal Messaging server as the destination for Integration Server to send or receive messages.

   For more information, see "Configure a JNDI Provider Alias" on page 42.

2. Configure a JMS connection alias that encapsulates the properties Integration Server requires to create a connection with the Universal Messaging server used as a JMS provider. Enable the JMS connection alias so that Integration Server can use the alias to obtain connections, send messages, and receive messages on behalf of the services and JMS triggers.

   For more information, see "Configure a JMS Connection Alias" on page 44.

For more information, see the *webMethods Integration Server Administrator's Guide*.

Configure a JNDI Provider Alias

The first time Integration Server starts, it creates a JNDI provider alias named DEFAULT_IS_JNDI_PROVIDER with predefined settings for establishing a connection to the local instance of the Universal Messaging server. If Universal Messaging is not installed in the same directory as Integration Server, then the DEFAULT_IS_JNDI_PROVIDER uses a provider URL of nsp://localhost:9000.

Configure the predefined JNDI provider alias named DEFAULT_IS_JNDI_PROVIDER to establish the Integration Server connection with the Universal Messaging server. To create a new JNDI provider alias, see "Create a JNDI Provider Alias" on page 42.

Create a JNDI Provider Alias

Use the following procedure to create an alias to a JNDI provider.

To create a JNDI provider alias

1. Open the Integration Server Administrator, if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JNDI Settings**.
4. Click **Create JNDI Provider Alias**.
5. Specify the following information for the JNDI provider alias:

In this field...	Specify...
JNDI Alias Name	The alias name that you want to assign to this JNDI provider.
Description	A description for this JNDI alias.
Predefined JNDI Templates	The JNDI template that you want to use. Select the Universal Messaging template. The JNDI templates provide information that you can use to complete alias configuration for a specific provider. **Note:** After you create a JNDI provider, Integration Server Administrator displays Current Settings as the value of the **Predefined JNDI Templates** field. This indicates that Integration Server uses the currently specified settings for the JNDI provider alias.
Initial Context Factory	The class name of the JNDI provider. The JNDI provider uses the initial context as the starting point for resolving names for the naming and directory operations. If you selected a predefined JNDI template, Integration Server displays the initial context factory for the provider.
Provider URL	The Universal Messaging server URL in the format nsp://*UM_host*:*UM_port* (for example, nsp://127.0.0.1:9000). If you are using a cluster of Universal Messaging servers, you can provide a list of comma-separated URLs to each server in the cluster. You can use semi-colon separated URLs in Universal Messaging 9.8 and later.
Security Principal	If basic authentication is configured, the principal name, or user name, supplied by 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 Credentials	If basic authentication is configured, the credentials, or password, that 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.

6. Click **Save Changes**.

Configure a JMS Connection Alias

A JMS connection alias specifies the information that Integration Server needs to establish an active connection between Integration Server and the JMS provider.

If you intend to use Universal Messaging server as a JMS provider, you need to configure one or more JNDI provider aliases before creating a JMS connection alias. For information about creating a JNDI provider alias, see "Create a JNDI Provider Alias" on page 42.

The first time Integration Server starts, it configures the default JMS connection alias named DEFAULT_IS_JMS_CONNECTION to connect to Universal Messaging by using the predefined JNDI provider alias named DEFAULT_IS_JNDI_PROVIDER. The DEFAULT_IS_JMS_CONNECTION connection alias is disabled by default.

Configure the DEFAULT_IS_JMS_CONNECTION alias or create a new JMS connection alias.

For more information, see "Configure the Default JMS Connection Alias" on page 44 and "Create a JMS Connection Alias" on page 45.

Configure the Default JMS Connection Alias

Edit the properties of DEFAULT_IS_JMS_CONNECTION alias to establish Integration Server connection with the Universal Messaging server, and make sure the JMS connection alias specifies JNDI lookup.

To configure the DEFAULT_IS_JMS_CONNECTION alias

1. Open the Integration Server Administrator, if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. In the JMS Connection Alias Definitions list, select the DEFAULT_IS_JMS_CONNECTION JMS connection alias.
5. In the **Create Connection Using** list, select **JNDI Lookup**.
6. Disable the DEFAULT_IS_JMS_CONNECTION JMS connection alias.
7. Click **Edit JMS Connection Alias**.
8. Edit the properties of the connection alias to establish Integration Server connection with the Universal Messaging server. For more information about the fields, see "Create a JMS Connection Alias" on page 45. Note that the **Connection Alias Name** field cannot be modified.

9. Click **Save Changes**.

Create a JMS Connection Alias

Integration Server uses a JMS connection alias to send messages to and receive messages from the JMS provider.

To create a JMS connection alias

1. Open the Integration Server Administrator, if it is not already open.
2. In the **Settings** menu of the Navigation panel, click **Messaging**.
3. Under JMS Configuration, click **JMS Settings**.
4. Click **Create JMS Connection Alias**.
5. Set the following **General Settings** for the JMS connection alias:

For this field...	Specify...	
Connection Alias Name	Name of the connection alias. Each connection alias represents a connection factory to a specific JMS provider.	
Description	A description of the JMS connection alias.	
Transaction Type	Whether sessions that use this JMS connection alias will be transacted.	
	Select...	**To...**
	NO_TRANSACTION	Indicate that sessions that use this JMS connection alias are not transacted.
	LOCAL_TRANSACTION	Indicate that sessions that use this JMS connection alias are part of a local transaction.
	XA_TRANSACTION	Indicate that sessions that use this JMS connection alias are part of an XA transaction.
Connection Client ID	The JMS client identifier associated with the connections established by this JMS connection alias.	

6. In the **Create Connection Using** list, select **JNDI Lookup**.

7. Do the following in the remaining fields under **Connection Protocol Settings**:

For this field...	Specify...
JNDI Provider Alias Name	The alias to the JNDI provider that you want this JMS connection alias to use to look up administered objects. For information about creating a JNDI provider alias, see "Create a JNDI Provider Alias" on page 42.
Connection Factory Lookup Name	The lookup name for the connection factory that you want to use to create a connection to the JMS provider specified in this JMS connection alias. Specify the Universal Messaging connection factory that you created when you set up your Universal Messaging environment or when migrated from Broker using the JNDI migration script.
Polling Interval (minutes)	The number of minutes between polling attempts. The polling interval must be a positive integer. The default value is 60 minutes. **Note:** This field is only available if you selected **Poll for changes (specify interval)**.

8. Under **Advanced Settings**, **Producer Caching**, and **Producer Retry**, specify the information for the JMS connection alias. For more information about configuring the advanced settings, producer caching, and producer retry for the JMS connection alias, see *webMethods Integration Server Administrator's Guide*.

9. Click **Save Changes**.

Optimize Configuration

Configure Universal Messaging with Optimize to enable these capabilities for Universal Messaging:

- Server discovery
- Data communication
- Asset discovery
- Server monitoring

If you use Infrastructure Data Collector with Optimize, you can monitor the Universal Messaging channels, queues, and datagroups that are running on the Universal Messaging server.

For more information about configuring Optimize, see *Administering webMethods Optimize*.

Perform these tasks to configure Universal Messaging with Optimize:

- Enable and configure the WmOptimize package. For more information, see "Enable and Configure the WmOptimize Package" on page 47.
- Specify Universal Messaging as the JMS server for Optimize. For more information, see "Specify Universal Messaging as the JMS Server for Optimize " on page 47.
- Enable Universal Messaging asset discovery. For more information, see "Add Universal Messaging Assets for Discovery" on page 48.

Enable and Configure the WmOptimize Package

Enable the WmOptimize package and configure Universal Messaging as the JMS provider to be used by the Analytic Engine.

To enable and configure the WmOptimize package on Integration Server

1. Open the Integration Server Administrator, if it is not already open.
2. Navigate to **Packages > Management**.
3. Locate WmOptimize package and click  for the WmPRT package.

   Verify that the WmOptimize package is enabled. A **Yes** appears in the **Enabled** column when a package is enabled. To enable a disabled package, click **No**.
4. Click **Settings** on the left hand navigation panel.
5. Click the **Edit Process Engine Settings** link.
6. Specify the Universal Messaging server URL in the **JMS Server URL** property. Change `localhost` to the correct host, and the port assignment to the correct port (if applicable), using the appropriate format for your JMS Server

   The default value of JMS Server URL is `nsp://localhost:9000`.
7. Click **Submit**.

Specify Universal Messaging as the JMS Server for Optimize

To specify Universal Messaging as the JMS Server for Optimize

1. Open the Integration Server Administrator, if it is not already open.
2. On the **Settings > Extended** page, click **Edit Extended Settings**.

3. For the watt.server.optimize.jms.server.url setting, specify the Universal Messaging URL. To add the watt.server.optimize.jms.server.url setting, perform one of these actions:

   - On the **Settings > Extended** page, click **Edit Extended Settings**. In the edit settings window, add the watt.server.optimize.jms.server.url setting. For example, watt.server.optimize.jms.server.url=nsp://localhost:9000.
   - On the **Settings > Extended** page, click **Show and Hide Keys**. Enable the check box next to watt.server.optimize.jms.server.url and click **Save**. Edit the watt.server.optimize.jms.server.url property setting and save the changes.

Add Universal Messaging Assets for Discovery

To add Universal Messaging assets for discovery

1. In My webMethods: **Navigate > Applications > Administration > Analytics > Infrastructure Components > Discovery**

2. On the Discovery page, click **Add Asset**.

3. In the Add Asset Discovery dialog, click the Down arrow to the right of the **Asset Type** field, and select the discovery type.

   Note:
   If you add an asset to your system or start an existing asset that was previously not running, you may need to refresh the Add Asset Discovery dialog in order for that asset to be displayed in the **Asset Type** list.

   The fields on the Add Asset Discovery dialog change to match those required by the selected discovery type.

4. Complete the fields on the Add Asset Discovery dialog.

Field	Description
***Data Collector**	Select the Infrastructure Data Collector to use for the discovery.
***Host**	Enter the host name or IP address for the discovery. The name must be unique.
***Server Port**	Enter the server port number for the discovery.
Username	Enter the User ID needed to log in to the asset.
***Protocol**	Enter the protocol used to connect to the Universal Messaging server being discovered. Supported protocols are nsp, nsps, nhp, and nhps. The default is nsp.

Field	Description
	To monitor a Universal Messaging server with SSL, select nsps as the protocol. Note that when using SSL, the Universal Messaging server must be configured to support SSL, and the Infrastructure Data Collector must be configured to use the appropriate key and truststores using the JNDI Configuration setting on the Configure Servers tab of the My webMethods Define Environments page. For more information about JNDI configuration, see the *webMethods Integration Server Administrator's Guide* guide.
Client Authentication	Select the client authentication type appropriate for the Universal Messaging server being discovered. Currently, the only available option is "None".

5. Click **OK** to add the specified asset, or click **Cancel** if you want to cancel the procedure without adding the discovery.

6. Click **Refresh** to update the information on the Discovery page.

Process Engine Configuration

Configure Process Engine to use Universal Messaging as the JMS provider to publish audit messages to Optimize.

For JNDI integration, configure a JMS alias named PE_NONTRANSACTIONAL_ALIAS. For information about creating a JMS alias, see "Configure a JMS Connection Alias" on page 44.

By default, the Process Engine uses a non-transactional JMS connection alias, PE_NONTRANSACTIONAL_ALIAS. PE_NONTRANSACTIONAL_ALIAS establishes a connection to Universal Messaging using the predefined JNDI provider alias DEFAULT_IS_JNDI_PROVIDER. If both Broker and Software AG Universal Messaging are installed on the local server, or only Universal Messaging is installed, then DEFAULT_IS_JNDI_PROVIDER is set to point to the Universal Messaging server. If you create this alias manually, you must reload the WmPRT package after you complete the alias creation.

Ensure that you complete the following to successfully run the migrated process models:

- Configure Process Engine to use Universal Messaging as the JMS provider.
- Configure migrated document types and triggers. For more information, see "Document Type and Trigger Migration" on page 32.

To open the Process Engine configuration settings page in Integration Server

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

2. Navigate to **Packages > Management**.
3. Locate the WmPRT package and click the Home Page icon.
4. On the Process Engine home page, click **Settings**.
5. On the Settings page, click the **Edit Process Engine Settings** link.

Configure Process Engine to use Universal Messaging as JMS Provider

1. Make sure the PE_NONTRANSACTIONAL_ALIAS JMS connection alias specifies the Universal Messaging server as the JMS provider.
2. In Universal Messaging Enterprise Manager, do the following:
   a. Create a new connection factory.

   Note:
   Do not perform steps b and c if you are using Universal Messaging 9.5.1 and later.

   b. Create these topics:
      - PEBroadcastTopic (the topic that the broadcast trigger listens on)
      - PERestartTopic (the topic that the restart trigger listens on)
   c. Update the security settings for the new connection factory and topics.
3. Ensure that the PE_NONTRANSACTIONAL_ALIAS JMS connection specifies the Universal Messaging server as the JMS provider.
4. Enable the PE_NONTRANSACTIONAL_ALIAS connection.
5. Set watt.prt.suppressCorrelationRetry property to `true`.
6. Reload WmPRT package in Integration Server.

Configure Optimize

To configure Optimize, see " Optimize Configuration" on page 46.

Configure Event Routing

For Universal Messaging 9.5.1 and earlier:

1. Specify the Universal Messaging server URL in the com.softwareag.eda.nerv.default.jms.provider property in the *Software AG_directory* /profiles/IS/configuration/com.softwareag.platform.config.propsloader/com.softwareag.eda.nerv.properties file.

   Note:
   In Universal Messaging server 9.6 and later, the Universal Messaging server URL is set by default in the following location: *Software AG_directory* /profiles/IS_default/configuration/com.softwareag.platform.config.propsloader/com.softwareag.eda.nerv.properties.

4 Troubleshooting

The following table lists information to help you troubleshoot post-migration Universal Messaging configuration scenarios:

Problem...	Cause...	Solution...

Error on JMS connection: Error retrieving JNDI context:
`javax.naming.`
`Communication Exception`
JNDI setup failed on RNAME
	The JNDI server cannot be reached or the URL specified in the associated JNDI alias is incorrect	Check if the URL in the associated JNDI alias is correct. If you are using a Universal Messaging cluster, ensure that the JNDI alias specifies all cluster nodes, either in the JNDI provider URL, or in the JNDI failover list

Error on JMS connection: Error creating connection factory:
`javax.naming.Naming`
`Exception`
Internal lookup failed for name
	ConnectionFactory not found in JNDI or lookup name in JMS alias is incorrect	Check if the given JNDI provider URL in the JNDI alias is correct. Check in the JNDI context on the specific server if a ConnectionFactory exists with the lookup name specified in the JMS alias. If using a Universal Messaging cluster, check that the channel *naming/defaultContext* is clustered correctly

Error on JMS connection: Error creating connection:
`javax.jms.JMSException:`
`com.pcbsys.nirvana.client.`
`nRealmUnreachableException`
	URL in ConnectionFactory cannot be reached	Check if the URL provided in the ConnectionFactory is reachable from the Integration Server machine. If Integration Server is on a different machine from where Universal Messaging is installed, ensure that the

Problem...	Cause...	Solution...
		ConnectionFactory URL does not include localhost
Error on JMS connection: Error retrieving JNDI context: `javax.naming.NoPermission Exception`	Client does not have the required permissions	Check if the Integration Server client has Access rights in the Universal Messaging server ACL and Subscribe rights in the *naming/defaultContext* channel ACL
Error on webMethods Messaging connection: Error initializing Universal Messaging session: `com.pcbsys.nirvana.client. nRealmUnreachableException`	URL in webMethods Messaging alias cannot be reached	Check if the Universal Messaging URL in the webMethods Messaging alias is correct and is reachable from Integration Server
Error enabling JMS trigger or sending JMS message: `javax.jms.Invalid DestinationException` Channel could not be found on the server	JNDI destination exists, but channel does not exist	Check that the URL in the ConnectionFactory used by the JMS alias refers to the same Universal Messaging server. If using a Universal Messaging cluster, check if the relevant topic is correctly clustered and available on all cluster nodes
Error enabling webMethods messaging trigger or publishing a webMethods messaging message: `com.pcbsys.nirvana.client. nChannelNotFoundException` Channel could not be found on the server	Channel does not exist	Synchronize the Integration Server document type with the Provider using Designer
Messages are not being delivered, or other error messages in a cluster	Cluster not configured correctly	Check if the Universal Messaging servers are correctly clustered. Check if the *naming/defaultContext* channels are correctly clustered. Check if the channels and queues are correctly clustered

Problem...	Cause...	Solution...
		If correctly clustered, the relevant icons appear orange in Enterprise Manager Check if a valid Universal Messaging cluster URL is configured in: - JNDI aliases in Integration Server - ConnectionFactories in Universal Messaging - webMethods Messaging aliases in Integration Server

A JNDI and JMS Migration Warnings

This section describes the warnings logged by the migration scripts. For additional information about Universal Messaging, see the Universal Messaging documentation.

JNDI Migration Script Warnings

Warning	**Ignoring Broker property "SSLEncrypted" with value "true"**
Cause	The migration script does not migrate the authentication related properties of the Broker connection factory. The SSLEncrypted property of the Broker connection factory is set to `true`.
Action	Configure the SSL encryption on the corresponding connection factory in Universal Messaging.

Warning	**Ignoring Broker property "SSLKeystore" with value <ssl_keystore_value>**
Cause	The migration script does not migrate the authentication related properties of the Broker connection factory. The SSLKeystore property of the connection factory is set in Broker.
Action	Configure the SSL keystore on the corresponding connection factory in Universal Messaging.

Warning	**Ignoring Broker property "SSLTruststore" with value <value>**
Cause	The migration script does not migrate the connection factory properties that are related to Broker authentication. The SSLTruststore property of the connection factory is set in Broker.
Action	Configure the SSL truststore on the corresponding connection factory in Universal Messaging.

Warning	**Ignoring Broker property "MarshalInClassName" with value <value>**
Cause	Universal Messaging does not support marshalling. The Broker-specific MarshalInClassName property of the connection factory is not migrated from the Broker. The MarshalInClassName property of the connection factory is set in Broker.
Action	Check the marshaling functionality in Broker and configure Universal Messaging as per your requirement for interoperability.

Warning	**Ignoring Broker property "MarshalOutClassName" with value <marshaloutclasname_value>**
Cause	Universal Messaging does not support marshaling. The Broker-specific MarshalOutClassName property of the connection factory is not migrated from the

	Broker. The MarshalOutClassName property of the connection factory is set in Broker.
Action	Check the marshaling functionality in Broker and configure Universal Messaging as per your requirement for interoperability.

Warning	**Ignoring cluster Brokers "[[<list_of_comma_separated_ cluster_Brokers>]]" as migrating to standalone UM**
Cause	The ClusterBrokers property of the Broker cluster connection factory is not migrated because it is difficult to identify: ■ Whether the number of Brokers and Universal Messaging servers are equal. ■ How load balancing is done using the cluster policy.
Action	Note the list of cluster Brokers in the warning message and make the required changes in the migrated cluster connection factory in Universal Messaging.

Warning	**Migrating to UM cluster in <Failover or Random> mode. Broker cluster policy is <policy_name>**
Cause	There is a mismatch due to one of these reasons: ■ The Broker cluster policy is not STICKY and the Universal Messaging server is in Failover mode. ■ The Broker cluster policy is STICKY and the Universal Messaging server is in RANDOM mode.
Action	Configure the Universal Messaging cluster to specify the list of servers in the cluster. Specify the list of server URLs separated by ";" or ",".

Warning	**Ignoring cluster connection factories "[< ClusterConnectionFactory_1, ClusterConnectionFactory_2>]" as migrating to standalone UM**
Cause	Universal Messaging does not support composite connection factory.
Action	Understand the usage of the composite cluster connection factory in Broker and configure Universal Messaging as per your requirement.

Warning	**Ignoring Broker property "SharedState" with value "false"**
Cause	The SharedState property of the Broker destination is disabled. The corresponding SharedDurable property in the Universal Messaging connection factory is enabled by default.
Action	Make sure you use this destination with a non-shared durable connection factory in Universal Messaging.

Warning	Ignoring Broker property "SharedState" with value " false"
Cause	The SharedState property of the Broker destination is disabled. The corresponding SharedDurable property in the Universal Messaging connection factory is enabled by default.
Action	Configure the SharedDurable property in Universal Messaging as per your requirement. Universal Messaging does not support shared state ordering.

Warning	Ignoring Broker property "PriorityOrdering" with value "true"
Cause	The PriorityOrdering property of a Broker destination is enabled. The corresponding SharedPriority property in the Universal Messaging connection factory is disabled by default.
Action	If the PriorityOrdering property is enabled in the Broker destination, enable the SharedPriority property in the Universal Messaging connection factory.

Warning	Ignoring Broker property "LocalOnly" with value "true"
Cause	The migration script does not migrate the LocalOnly property of the Broker topic. The LocalOnly property is enabled in Broker.
Action	Configure Universal Messaging to receive only the messages published by the local publisher, if required.

Warning	Ignoring Broker property "DeadLetterOnly" with value " true"
Cause	The migration script does not migrate the DeadLetterOnly property of the Broker topic. The DeadLetterOnly property is enabled in Broker.
Action	Configure Universal Messaging to subscribe to dead letters, if required.

Warning	Migration not done for <sub_context_name> with value <subcontext_value>
Cause	The migration script does not migrate the subcontext JNDI entries of Broker.
Action	Create the folders manually and move your destinations to the target Universal Messaging to achieve the required structure.

Warning	Migration not done for String <string_name> with value <string_value>
Cause	The migration script does not migrate the string JNDI entries of Broker.

JMS Migration Script Warnings

Warning	**Ignoring Broker client group "JMSClient" property required_encryption" with value "ENCRYPT_LEVEL_ENCRYPTION"**
Cause	The migration script does not migrate any authentication related information from Broker. Encryption is enabled for the client group in Broker.
Action	Configure authentication as per your requirement in Universal Messaging.

Warning	**Ignoring Broker client group "JMSClient" property "access_label_required" with value "true"**
Cause	The migration script does not migrate any authentication related information from Broker. Access label, an advanced security feature of Broker is enabled for the client group.
Action	Configure authentication as per your requirement in Universal Messaging.

Warning	**Ignoring Broker client group "JMSClient" property "ClientGroupACL" with value "Users: [CN=brokerserver,O=webM,ST= CA,C=US], AuthNames: [emailAddress=dtung@webmethods.com,CN= webMCA,O=webM,ST=CA,C=US]"**
Cause	The migration script does not migrate any authentication related information from Broker. ACL is set for the client group in Broker.
Action	Configure authentication as per your requirement in Universal Messaging.

Warning	**Ignoring Broker document type property "Validation" with value "FULL or OPEN"**
Cause	The validation property of the document type specifies whether Broker must validate the document type instances. The validation property set to "FULL" or "OPEN" specifies that Broker must validate even the fields that are not defined in the document type.
Action	Universal Messaging does not support message validation at the server side. Configure message validation as per your requirement in your messaging solution.

Warning	**Ignoring Broker document type property "Storage type" with value "VOLATILE"**
Cause	The storage type property of the document type in Broker determines whether the instances of the document type are stored in memory or disk. The channel type is Mixed in Universal Messaging by default, whereas the storage type of the document type in Broker is set to VOLATILE.
Action	If the Broker storage type is set to VOLATILE, configure the channel type to transient in Universal Messaging.

Warning	**Ignoring Broker document type field(s) "[name, host, type, version, client group, info]"**
Cause	The Fields property of the Broker document type is not migrated. The Fields property contains data that client applications use to exchange information. The Fields property might contain a single value, a sequence of values of the same type, or a structure containing values of different types.
Action	If the document fields configured in Broker have functional implications on Universal Messaging, configure Universal Messaging as per your requirement.