Universal Messaging Release Notes

Version 10.1

October 2017

This document applies to Universal Messaging Version 10.1 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.

Document ID: NUM-RN-101-20171103

Table of Contents

1 About this Guide

The *Release Notes* for Universal Messaging describe the changes introduced with the current product release.

Documentation roadmap

Universal Messaging provides documentation in the following formats:

- HTML
- PDF

The following table describes the different guides that are available.

Title	Description
Release Notes	Describes new features and changes since the previous release.
Installation Guide	Describes how to install the product.
Concepts	Describes the underlying concepts of the Universal Messaging product.
Administration Guide	Describes how to perform administration tasks for your Universal Messaging environment, by using either using a graphical user interface or an API.
Developer Guide	Describes how to develop client applications for enterprise, mobile clients or web clients.
Reference Guide	Provides programmer-level API documentation for the supported APIs and languages.

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.

Contacting customer support

If you have an account, you may open Universal Messaging Support Incidents online via the eService section of Empower at https://empower.softwareag.com/. If you do not yet have an account, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

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

Supported Product Releases and Platforms

Supported Product Releases

In addition to the current product release, older versions of the product remain may continue to be supported for a certain period of time.

For a list of product releases and their support periods, refer to the document *Release Availability* that is available from the following web page: http://documentation.softwareag.com/webmethods/universal_messaging/umessaging_vers.htm.

Each new release of Universal Messaging has its own *Release Availability* document. Be sure to consult that document for details about supported releases.

Supported Platforms

Universal Messaging runs on the platforms listed in the *Supported Platforms* document that is available from the following web page: http://documentation.softwareag.com/webmethods/universal_messaging/umessaging_vers.htm.

Each new release of Universal Messaging has its own *Supported Platforms* document. Be sure to consult that document for details about supported versions of operating systems, web browsers, etc.

2 What's New In Universal Messaging 10.1

Universal Messaging 10.1 is the successor of Universal Messaging 10.0.

Universal Messaging 10.1 includes new features, enhancements, and changes as described in the following topics.

Updated and New Functionality in the client API for C#

- **Shared durable subscriptions now supported in client API for C#**

  The client API for C# now supports *shared durable subscriptions*. The client API for C# functionality used in previous product releases for creating named objects and named objects with priority has now been deprecated.

  The new API provides different public methods for interaction. For operations like creating, retrieving, deleting or unbinding a durable, the nDurableManager must be used. Every channel has a durable manager associated with it.

  For information how to use the new functionality, refer to the topic *Using Durable Objects* in the C# section of the *Developer Guide*. See also the section *Named Objects, Shared Named Objects and Serial Named Objects* in the *Concepts* guide.

- **Serial durable subscriptions now supported in client API for C#**

  The client API for C# now supports *serial durable subscriptions*.

  With a serial durable subscription, multiple subscribers can hold a subscription to the same named object and all the subscribers will process events in a serial manner.

  For information about the new functionality, refer to the topic *Using Durable Objects* in the C# section of the *Developer Guide*. See also the section *Named Objects, Shared Named Objects and Serial Named Objects* in the *Concepts* guide.

- **New public interface for committing and rolling back events**

  A new public interface has been added for committing and rolling back events. The methods are defined for the nDurable instance and the usage of the old API, e.g. calling ack() or rollback() on the received event's nConsumeEvent object, is not recommended since it does not fully support individual acknowledging and rolling back. To be able to apply these operations on a single event and not only on consecutive event IDs is a significant importance for the shared types.

- **Basic Authentication now supported in C# client API**

  It is now possible to use SASL (e.g. plain text) authentication for the client API for C#. Previously, this functionality was only available using the client API for Java.

  For details, refer to the section *Basic Authentication* in the C# section of the *Developer Guide*.

Serial durable subscriptions now supported in client API for Java and as an extension to the API for JMS

The client API for Java now supports serial durable subscriptions. The same functionality has been added as an extension to the Universal Messaging API for JMS.

With a serial durable subscription, multiple subscribers can hold a subscription to the same named object, and all the subscribers will process events in a serial manner.

For information about the new functionality, refer to the topic *Durable channel consumers and named objects* in the Java section of the *Developer Guide*. Refer also to the section *Named Objects, Shared Named Objects and Serial Named Objects* in the *Concepts* guide.

Periodic logging of the realm server status

The logging feature has been extended to allow the status of the realm server to be reported in the server log at regular intervals. The status includes metrics such as the amount of memory currently in use for active events, the amount of disk space in use, CPU load, number of active connections, total bytes sent and received.

For details, see the section *Periodic Logging of Server Status* in the *Concepts* guide.

New and Enhanced Command Central Capabilities

Command Central can now be used to configure Universal Messaging zones and security groups.

Command Central now supports enhanced port configuration options.

You can now configure round-robin JMS Connection Factories using Command Central, allowing messages to be distributed evenly across multiple Universal Messaging realms or clusters. For more information, see the Universal Messaging Administration Guide.

These capabilities can be accessed using the Command Central web user interface, command-line interface, REST API, and composite templates.

Changed handling of missing operator in filter expression

In v10.0 it was permissible to omit a logical operator between two selector clauses in a filter expression. In such cases, the omitted operator was treated implicitly as an "AND" operator. For example, instead of the correct form:

```
( Item1 = 'ABC' ) AND ( Item2 in ( 'Invoicing', 'Pending' ) )
```

it was possible to state:

```
( Item1 = 'ABC' ) ( Item2 in ( 'Invoicing', 'Pending' ) )
```

In 10.1, omitting the logical operator will be flagged as an error. Therefore, when you upgrade to 10.1, ensure that you modify your filter expressions accordingly.

Pause Publishing

The new *pause publishing* feature allows all client publishing to the server to be paused, across channels, queues and data groups. This pause will not affect the administration API, inter-cluster communication or joins.

The feature is activated by setting the server configuration property `PauseServerPublishing` to true. Then, clients trying to publish or commit will receive `nPublishPausedException`. Exception handlers from prior to the current product version will handle this as `nSessionPausedException`, which the new `nPublishPausedException` extends.

Information about this feature is available in the section *Pause Publishing* in the *Concepts* guide.

If client applications using APIs from previous product versions are used to publish events to a v10.1 server, the following changes will be observed:

Old API (prior to 10.1)	Behavior publishing to a v10.1 server
Native API for Java	If a client API from a previous product version is used to connect to a 10.1 server, Java native clients will receive `nBaseClientException` instead of `nPublishPausedException`.
API for JMS	The behavior here is the same as for 10.1, with the difference that the root JMSException will be `nBaseClientException` instead of `nPublishPausedException`.
API for C++/C#	The APIs for C++ and C# will throw `nUnexpectedResponseException` when publishing is paused.

Updates for MQTT

Configuration Properties for MQTT

A new set of realm configuration properties for MQTT has been added.

For the list of MQTT configuration properties see the topic *Realm Configuration* in the Enterprise Manager section of the *Administration Guide*.

Support for clustered channels

MQTT support for clustered channels has been added.

General improvements

Non-functional aspects such as performance, availability and scalability have been improved.

Server JAAS Authentication with Software AG Security infrastructure component

Universal Messaging can now use the Software AG Security Infrastructure component (SIN) to provide server JAAS authentication capabilities. The SIN component provides a variety of options for using different authentication back-ends and implementing flexible authentication scenarios.

The SIN module is now the default LDAP module for use with Universal Messaging.

SIN modules are delivered with the Universal Messaging distribution and are available on the Universal Messaging server classpath.

For details, see the section *Security* in the *Concepts* guide.

Configuring Universal Messaging for use with IBM WebSphere Application Server

The Universal Messaging installation contains a product-specific generic resource adapter for JMS. Universal Messaging can be configured to work with IBM WebSphere Application Server via this adapter.

For details, see the section *Resource Adapter for JMS* in the *Developer Guide*

Microsoft Edge is a supported Web browser for Javascript Communication Drivers

The list of web browsers that are supported for the Javascript communication drivers has been extended to include Microsoft Edge for several of the drivers.

For details, see the section *Communication Drivers* in the *Developer Guide*.

Corrected DLL names in the product documentation

Some sections of the documentation referred to the DLL files "Nirvana DotNet.dll" and "Nirvana Silverlight.dll" by the wrong names "Universal Messaging DotNet.dll" and "Universal Messaging Silverlight.dll" respectively. These have been corrected.

These DLLs are mentioned in the C# and VBA sections of the Developer Guide.

New Java system property "Nirvana.sasl.client.enablePrehash"

The new Java system property `Nirvana.sasl.client.enablePrehash` specifies whether to prehash the supplied password when using the CRAM-MD5 or Digest-MD5 mechanisms.

For details, see the section *Client-side Authentication* in the Java section of the *Developer Guide*.

Removed and deprecated features in 10.1

The following Universal Messaging features are now deprecated or have been removed in Universal Messaging 10.1:

- **Deprecated Client API Support for Microsoft Silverlight**

The Universal Messaging client API for Microsoft Silverlight is deprecated and will be removed from the product distribution in the next official release.

- **Removed realm configuration parameter "QueuedSharedDurableFilterBound"**

  The realm configuration parameter `QueuedSharedDurableFilterBound`, which was introduced in Universal Messaging 10.0, has been removed. This boolean parameter specified whether or not to bind a shared durable to the event filter of its latest user.

  The behavior in v10.1 without this parameter is the same as the 10.0 behavior when this parameter was set to "true", namely: once the shared durable is bound to a filter, any subsequent subscriptions that mismatch the filter will replace the filter and an asynchronous exception will be thrown to existing subscribers.

- **Removed realm configuration parameter "OutputBlockSize"**

  The realm configuration parameter `OutputBlockSize` has been removed.

- **Deprecated methods in C# Client API**

  The methods ack() and rollback() on the nConsumeEvent object of the received event is deprecated.

  See the section "Updated and New Functionality in the client API for C#" on page 9 above for information on the new API features for committing and rolling back events.

- **Deprecated Client API Support for Windows Phone**

  The Universal Messaging client API for Windows Phone is deprecated and will be removed from the product distribution in the next official release.

- **Deprecated Universal Messaging Directory Backend Authentication**

  The Internal User Repository and LDAP directory backend providers that were in use in Universal Messaging v10.0 for authentication have been deprecated. The directory backend providers are being replaced by the new functionality described in the section "Server JAAS Authentication with Software AG Security infrastructure component " on page 12 above.

- **Deprecated Administration APIs for C# and C++**

  The Administration APIs for C# and C++ are deprecated and will be removed from the product distribution in the next official release.

3 What's New In Universal Messaging 10.0

Universal Messaging 10.0 is the successor of Universal Messaging 9.12.

Universal Messaging 10.0 includes new features, enhancements, and changes as described in the following topics.

Durable subscribers can be browsed in Command Central

Command Central can now be used to browse individual messages in durable subscribers in a Universal Messaging server instance, including messages waiting to be received by Integration Server triggers. You can view the size of individual messages and some properties of each message. You can also view the payload of string messages. All or individual messages can be deleted from the durable subscriber (for some durable subscriber types).

Enhanced Command Central support for Universal Messaging

Command Central can now be used to configure realm ACLs for Universal Messaging, allowing you to control which users can perform what actions at the realm level. You can also use Command Central to configure Java system properties for Universal Messaging. For Universal Messaging clusters, Command Central now displays whether each running instance is a master or slave in the cluster.

These capabilities can be accessed using the Command Central web user interface, command-line interface, REST API, and composite templates.

Utility for migrating webMethods Broker gateways configurations to Universal Messaging

For webMethods Broker users who have configured territories and gateways and have large numbers of document types (topics) configured in the gateway connections, migrating these to equivalent remote joins in Universal Messaging previously involved a lot of manual effort. Now, you can use a new utility that can read the gateway configuration from webMethods Broker and automatically establish the same remote joins in Universal Messaging.

Servers can be instructed not to restrict incoming messages from specific clients in low memory situations

A client session can request to bypass the existing server-side low-memory throttling. This is intended to allow administrators or administrative services to continue to be able to connect in order to resolve the low-memory situation.

JNDI assets can now be stored in existing (non-Universal Messaging) JNDI providers

Users who wish to use a JNDI provider for binding JNDI assets such as Connection Factories and Destinations can now store their Universal Messaging JNDI assets in a

JNDI provider of their choice. This capability is only available through the respective API.

Cluster side-by-side upgrades to new machines are easier to manage

You can now use a migration utility to migrate cluster configurations to machines that are on different hosts from the original cluster.

Shared durables re-architected to improve efficiency and robustness

The implementation of shared durables has been re-written to improve efficiency. The new implementation performs tracking of durable objects in-place on the channel, rather than relying on additional internal stores. This approach saves memory and reduces the complexity of the solution, improving robustness. A new API has been developed to manage durable objects through the Universal Messaging native API. The Universal Messaging JMS library has also been updated to utilize the new durable implementation.

Configuration using Enterprise Manager is simplified

Enterprise Manager hides a number of configuration settings behind an **Advanced** button. These hidden settings are ones that are rarely recommended to be modified.

Additional tools are now available showing how to use the API with datagroups and illustrating how to publish and subscribe

Tools are provided that allow management of datagroups and illustrate how to use publish and subscribe.

Robustness improvements for installations using the default out-of-the-box configuration

The default configuration for new installations has been reviewed and tested to ensure Universal Messaging is robust.

Named object ID is a concatenation of client ID and durable subscriber ID

Named objects IDs are now a concatenation of the client ID and the durable subscriber ID. Previously Universal Messaging just used the durable subscriber ID provided by the client.

Removed and deprecated features in 10.0

The following Universal Messaging features are now deprecated or have been removed in Universal Messaging 10.0:

- The **SharedDurableFilterBound** option, found in the **Durable Config** group in the administrative API and Enterprise Manager/Command Central, has been migrated to be a System Property (**-DQueuedSharedDurableFilterBound**). This new system property is deprecated and will be removed in a future release.

4 What's New In Universal Messaging 9.12

Universal Messaging 9.12 is the successor of Universal Messaging 9.10.

Universal Messaging 9.12 includes new features, enhancements, and changes as described in the following topics.

Durable subscribers monitoring and API improvements

Enterprise Manager has improved monitoring of durable subscribers and is now able to display more details for the durable subscribers, including details about the connections currently be used, the EIDs and the number of events outstanding in the queues. This information can now also be accessed via the administration API.

The client API for durable subscribers and named objects has been redesigned to improve performance, robustness and usability. The new durable subscribers API, available from the client API, maps to the existing durable subscribers functionality.

New and enhanced Command Central capabilities

You can now use Command Central to add, edit, delete, administer, and monitor channels (topics) and queues. In addition, you can monitor durable subscribers to easily detect and identify issues such as stalled triggers or processing backlogs.

These capabilities can be accessed using the Command Central web user interface, Command Central command-line interface, and REST API.

The Command Central web user interface now provides the following capabilities:

- Create and delete Universal Messaging server instances.
- Search for JNDI entries, channels, and queues.
- View, create, edit, and delete access control lists (ACLs).
- View create, edit, and delete joins for a channel or a queue.
- Delete durable subscribers.

Improved handling of low memory situations

New methods for protecting against out-of-memory situations have been introduced to increase the robustness of Universal Messaging under heavy load.

The "event usage" metric provides information on memory currently in use by on-heap events. This includes current on-heap event memory usage, the maximum memory currently available to the JVM, and the percentage of on-heap memory currently in use. These statistics enable monitoring of the current memory usage, allowing action to be taken accordingly.

Universal Messaging servers can now throttle producing connections while processing their events. At predefined, configurable thresholds of on-heap event memory usage, producer connections are throttled, enabling consumers to reduce the number of events on the connections while they are throttled. Connections are more strictly throttled as memory usage rises, helping to prevent out-of-memory situations.

Round-robin message publishing using connection factories for JMS

Horizontal scalability improvements have been introduced with the API for JMS, now allowing the configuration of round-robin connection factories. These factories allow clients to publish messages in a round-robin fashion, so that one message or transaction gets published to the first realm node or cluster, the next message to the next realm node or cluster, and so on.

These connection factories for JMS have the following limitations:

- Event consumption is not supported through these factories so, for example, message listeners cannot be registered and consumers cannot be created via the sessions created from these connection factories.
- The sessions created through these connection factories do not support distributed (XA) transactions.

For more information consult the JMS-related section of the product documentation.

Logging capabilities enhancements

Support has been added for utilizing the third party logging frameworks Logback and Log4J 2. Both of these testing frameworks offer improved throughput performance when compared to the existing Flogger engine.

Log file entries are now categorized by the component which generated the entry, e.g. Cluster Communications, Joins, etc.

Improved futureproofing for Universal Messaging clients

The client API is now officially supported for use with newer versions of the Universal Messaging server, which means that the 9.12 client API will be supported for use with future versions of UM server.

The client API has been extended in this release with features that were previously only in the administration API (which is not supported for use with newer versions of the server).

ACLs can now be set at store-creation time via the client APIs for Java and C++. This allows basic ACL control for stores without needing to use the administration API.

Setting ACLs at store-creation time has been a typical use for the administration API. These changes allow the client API to be used in a greater number of use cases. The client API is more lightweight than the administration API, and therefore switching to the client API can increase overall system performance and consumed bandwidth.

HTTP drivers support checking of "Origin" headers

The HTTP/WebSocket drivers have been updated to process the Origin header field according to standards proposed in RFC-6454, RFC-6455 and the W3C Cross Origin Resource Sharing document (https://www.w3.org/TR/cors).

Any nhp/nhps interfaces may have to have their CORS Allowed origins (located under the nhp/nhps Interface -> Javascript tab in Enterprise Manager) altered if an HTTP request has the Origin header field set. Previous versions of Universal Messaging had default values of "localhost, 127.0.0.1" assigned to the CORS Allowed origins field, and would process only host names as values to this field. The current W3C standards now expect any origin to be of the form "<scheme>://<host>:<port>"; for example, "localhost" is an incorrect value, while "http://localhost:11000" is a properly formatted value. The exception is a single value of "*", which indicates that all hosts are permitted access; note that the processing of this value has not changed with the update, and is now the default value in the CORS Allowed origins field whenever an nhp/nhps interface is created.

In addition, support for matching "http://example.com" and "http://example.com:80" as origins (as documented in RFC-6454) is currently not supported. You will need to explicitly white list hosts with *:80 as potential origins (if needed) in addition to others.

Warning of the effects of editing stores

When stores are edited, Universal Messaging deletes and recreates the store and this can disrupt active subscriptions. Enterprise Manager has been updated to display a warning message that the store will be recreated, before a channel edit is performed.

nInterfaceTool extended to allow editing of additional interface settings (e.g. autostart)

The nInterfaceTool has been extended to provide additional capabilities. For example, it now allows you to set interfaces to automatically start when the server starts.

Docker 1.10 support

The Docker kit for Universal Messaging now supports Docker 1.10.

Python and iOS client libraries included in installation

The client libraries for Python and iOS are now included as part of the installation.

UM-tools "runner" is installed as part of the "Template applications" module

The um-tools runner is now part of the installer "Template applications" module rather than the "Server" module. This allows you to install the um-tools runner without installing the server.

Updated version of OpenSSL

Universal Messaging now uses OpenSSL 1.0.2 instead of the previous version 1.0.1.

Platform changes in 9.12

Universal Messaging 9.12 runs on the platforms listed in the *Supported Platforms* document that is included in the Universal Messaging 9.12 documentation set. You can find this documentation set in the Software AG Documentation web site as described in the section "Supported Product Releases and Platforms" on page 7.

Check the above mentioned *Supported Platforms* document for details about the newer platform versions supported by Universal Messaging 9.12.

Removed and deprecated features in 9.12

The following Universal Messaging features are now deprecated or have been removed in Universal Messaging 9.12:

- Support for Flex has now been removed from Universal Messaging. Flex is a rich internet application (RIA) language which allows the development of complex web applications that run inside a browser. Flex is no longer supported by Adobe Systems Incorporated and has been transferred to the Apache Software Foundation.
- Support for P2P has now been removed from Universal Messaging. The API for P2P is a legacy API within Universal Messaging. It allows stream-based communication between two clients mediated by the Broker. This messaging system is no longer useful in the light of more recent and modern paradigms such as DataGroups.

5 What's New In Universal Messaging 9.10

Universal Messaging 9.10 is the successor of Universal Messaging 9.9.

Universal Messaging 9.10 includes new features, enhancements, and changes as described in the following topics.

Docker support

A Universal Messaging Packaging Kit for Docker is now part of the standard Universal Messaging installation on Linux. The kit was previously available only on TECHcommunity.

The Universal Messaging Packaging Kit can be found here:

```
<SAG Install Folder>/UniversalMessaging/server/<UM Server Name>/bin/docker
```

The kit includes the following Docker tools:

- Dockerfile for creating a Docker image from the Universal Messaging installation on Linux.
- Samples showing how to start the Universal Messaging server from the Docker image and run the sample applications of Universal Messaging within the image.
- The TradeSpace demo application shows how to use 'docker-compose' to set up and run the TradeSpace demo of Universal Messaging.

Optimized persistent store

There is a new form of persistent store, enabled by setting the spindle size to a value greater than zero. This store format uses multiple files for each channel or queue and removes the overhead of Universal Messaging which has to "perform maintenance" on them. This new mechanism also allows stores to grow without any restriction on the JVM heap. The standard persistent store mechanism keeps an in-memory index which grows each time a message is added.

Improved Handling of Maximum Message Size

In previous releases, Universal Messaging restricted the maximum message size that the server will read in by the MaxBufferSize configuration property. Exceeding this value would cause the connection to be disconnected, but the message had already been sent over the network and the reason for the disconnection was not obvious to the client. This check by the server remains but Universal Messaging now has a client side check so that the message is rejected before it is sent over the network and the user can handle the exception.

Clients can “follow the master node”

Clients can now be enabled to "follow the master" in a cluster. The client will initially connect to a server in the cluster but if that server is not the master, the client will be redirected to the master node. Connecting to the master node can provide better performance in some use cases.

Transactions over AMQP

The AMQP specification defines a number of transactional operations. Universal Messaging now supports AMQP transacted operations so that client applications can perform transactional work when communicating with the realm server over AMQP. For example, if an application communicates to the realm server using a JMS AMQP client library (e.g. Apache Qpid JMS client) it can take advantage of the local transaction functionalities defined in the JMS specification.

Universal Messaging does not currently support the AMQP Transactional Acquisition operation. However, this sets no limitations on using JMS transactions over AMQP.

Configuration profiles in Installer

It is now possible to select different configuration profiles using the Software AG Installer. We have provided two configurations: one tuned for typical webMethods use cases and one tuned for standalone use cases.

JNDI asset configuration in Command Central

Command Central now supports the creation and maintenance of JNDI connection factories, queues, and topics for Universal Messaging. These assets can be managed through the Command Central web user interface, command line interface, REST API, and within templates.

Enterprise Manager handles realm name conflicts more gracefully

You can no longer connect to a realm with the same name as the realm to which you are already connected. Enterprise Manager now shows the host and port in the realm tree to make it clear which realm is being referenced.

Additional sample applications

New sample applications are included that can be used to:

- Create clusters
- Create server interfaces
- Manage security groups

Additional documentation

Various items have been added to the product documentation:

- The documentation has been updated with a guide helping you configure your JVM in order to support FIPS 140-2.
- The documentation now includes best practice guidelines helping you to configure Universal Messaging to have separate interfaces for client and cluster communication. The guide describes how to restrict regular client communication but allow administrators to connect when a cluster is forming.
- Documentation is also provided describing the API's setMessageType() method that allows a client to convert a JMS message back to its original type.

Removal of Standalone Installer

Universal Messaging can no longer be installed using its own standalone Installer. It can now only be installed using the standard Software AG Installer or Command Central.

6 What's New In Universal Messaging 9.9

Universal Messaging 9.9 is the successor of Universal Messaging 9.8.

Universal Messaging 9.9 includes new features, enhancements, and changes as described in the following topics.

Automatic creation of JNDI entries during JMS migration from webMethods Broker

For users who were using webMethods Broker connections for JMS configured using Native webMethods API instead of JNDI Lookup, it was necessary to manually create JNDI destination entries for migrated queues and channels. Now, the Broker-to-Universal Messaging migration utility for JMS provides the option to auto-generate relevant JNDI destination entries for migrated queues and channels.

Realm properties configurable using Command Central

Command Central now provides support for configuration of Universal Messaging realm properties using the Command Central web user interface, command-line, or templates.

JMS Resource Adapter

Universal Messaging now provides a pre-packaged JMS Resource Adapter for easy use in 3rd party application servers.

Enterprise Manager enhancements

The following usability improvements are now available in Enterprise Manager:

- When creating or editing JNDI Connection Factories, the user can now more easily select, view or change whether the Connection Factory has the Shared Durable flag set. This flag defines whether multiple client connections are allowed to concurrently read from JMS Topic Durable Subscriptions.
- When viewing Named Objects (also known as Durable Subscriptions) associated with a channel, Enterprise Manager now shows the number of messages not yet retrieved by the subscriber.
- Removal of the login section at the top of the JNDI panel provides a streamlined user interface for JNDI editing.

AMQP 1.0 Support

Universal Messaging now supports a significant part of the AMQP 1.0 standard wire protocol for messaging, specifically the parts that are exposed through the JMS API.

The following capabilities are supported:

- TCP and TLS (alternative NOT negotiated) transport

- AMQP PLAIN & AMQP over SASL
- Publishing to Topics and Queues (non-transactional)
- Temporary Queues
- Keep Alives
- Synchronous and Asynchronous consuming on topics and queues (non-transactional)
- Flow control
- Durable subscribers
- Interoperability with UM publishers and subscribers (using any of the available client APIs)
- Interoperability with existing AMQP JMS clients (SwiftMQ, Apache QPID and Apache Legacy QPID)
- Some interoperability with MQTT clients

Note: AMQP support is only available with the fully-featured Universal Messaging license (NUMWF or NUMTF).

Adapter Notifications Support for non-default webMethods Messaging aliases

Adapter notifications generate publishable document types, which were previously always bound to the default webMethods Messaging connection alias. Now, the developer can select which specific messaging connection to use for each generated document type.

Testing capabilities support Universal Messaging

Developers can now use two testing features of Software AG Designer with Universal Messaging:

- Publishable document types can be sent from Software AG Designer to Universal Messaging using the Run As Publishable Document option.
- webMethods Messaging Triggers subscribing from Universal Messaging can be tested using the Run As webMethods Messaging Trigger option.

Support of authenticated connections for deployment of assets to Universal Messaging

webMethods Deployer now supports deployment of assets to Universal Messaging servers that are configured to enforce authentication. Both basic authentication (username/password) and certificate-based authentication are supported.

New umStorage API for working with storage types

The Universal Messaging server has a highly optimized I/O subsystem for efficient serialization of data to and from disk. The umStorage API (Java only) abstracts this layer from the server so it can be used directly by applications. This simple API allows the

user to construct any of the standard Universal Messaging storage types (Mixed, Off Heap, Reliable etc.) and easily serialize/deserialize data.

New ordering policy for rolled back messages on queues

When a transactional queue consumer rolls back a message, it needs to be put back onto the queue. Prior to 9.9 this message would be simply republished to the end of the queue. In 9.9 we have changed the default behavior so that the message is re-added to the front of the queue.

Per-session SSL settings

It is now possible to specify custom SSL settings per nSession rather than per JVM, e.g. key store, trust store.

JMS resource adapter support

As part of the install we now ship a resource adapter for JMS. Users with existing JMS applications that run on an application server such as JBoss can now seamlessly switch to Universal Messaging as a JMS provider.

Support for dynamically updating Google Protocol Buffer schema

Universal Messaging allows you to attach a Google Protocol Buffer schema to a channel/queue but until 9.9, changing the schema would require deleting and recreating the channel. A channel edit that only changes the schema is now supported without recreating the channel.

Usability improvements for shared named objects

It is now possible to monitor the number of messages outstanding for a shared named object. Due to the architecture of shared named objects it was not previously clear how many events were being stored waiting to be consumed. This information is now available in the API and also visible in the Enterprise Manager.

7 What's New In Universal Messaging 9.8

Universal Messaging 9.8 is the successor of Universal Messaging 9.7.

Universal Messaging 9.8 includes new features, enhancements, and changes as described in the following topics.

Automatic Interest Propagation Using Zones

Universal Messaging now allows the configuration of Zones. Realm servers in the same Zone inform each other about local subscribers to channels/topics, so that other members of the Zone can forward messages to local subscribers when required. This minimizes unnecessary wide area network (WAN) traffic by only forwarding messages when there is a known interested party on the remote Realm server.

Also, users can configure Zones from the Enterprise Manager or by using a newly introduced section of the Admin API. This feature is the equivalent of webMethods Broker Territories

Basic Authentication with Admin API and Enterprise Manager

The Universal Messaging Admin API and Enterprise Manager support the passing of basic authentication credentials when a connection is established. This provides a more secure configuration of the Realm and improved authentication of administrative clients.

Realm XML Export Includes JNDI Assets

XML export of a Realm server now includes the option to export JNDI assets. These can be Connection Factories, Topic Connection Factories, Queue Connection Factories, XA Connection Factories, JNDI Topics and JNDI Queues.

These are used for deployment using Deployer.

New API for Zone Management

The new API com.pcbsys.nirvana.nAdminAPI.nRealmNode provides zone management functionality through the Universal Messaging Administration API, allowing users to create a zone, join a realm to a zone, remove a realm from a zone, and obtain realm zone information:

- public Zone createZone(String zoneName)
- public void joinZone(Zone zone)
- public void leaveZone(Zone zone)
- public Zone getZone()

MQTT 3.1.1 Support

Support for the standardized MQTT protocol version 3.1.1 has been added in addition to the legacy 3.1 version. Additional features now include automatic topic creation, authentication support as well as retained messages and session persistence.

Removal of MQTT Virtual Payload types

MQTT virtual payload type support has now been removed for reasons of compliance to the standard. All MQTT messages are now being handled as JMS bytes messages.

8 What's New In Universal Messaging 9.7

Universal Messaging 9.7 is the successor of Universal Messaging 9.6.

Universal Messaging 9.7 includes new features, enhancements, and changes as described in the following topics.

C++ Client Improvements

New features have been added to the existing C++ client for webMethods Universal Messaging to make the client more robust and improve its performance.

Brokerless API Renamed to umTransport API and Supported in C++

The existing Brokerless API for lightweight client-client communication has been renamed to umTransport API.

This umTransport API is now available for C++ clients, in addition to the existing support for Java clients. The C++ does not presently support asynchronous communication between clients and only supports TCP sockets and TCP secure sockets as a communication transport.

Performance Improvements

The following enhancements were made to improve performance:

- New Paged channel types provide the ability to use high-speed, off-heap memory that stores events in persisted memory mapped files. This new channel type increases performance over other persisted channel types.
- JavaScript drivers have benefited from various performance improvements.
- Interest propagation: When joining channels, it is now possible to manage the interest within the channels for sending and receiving events as part of the join. This can reduce the amount of bandwidth used by only forwarding events from channels where there are subscribers to a channel of the same name on the remote realm or cluster.

Protocol Buffers Administration API

A new Java API provides the ability to manage the Protocol Buffer (Protobuf) configuration on the server. Integration Server uses this API to synchronize document types and support server-side filtering.

ninistancemanager Command Line Tool No Longer Installed within the Default Instance

The ninstancemanager tool is now installed separately from the default instance during installation. This provides more flexibility during installation and removes the dependency that the initial default instance must always be present.

Additional Support for Universal Messaging in Command Central

The following Universal Messaging administration tasks can now be performed in Command Central:

- User management
- Cluster configuration
- Memory configuration settings
- Log file access

9 What's New In Universal Messaging 9.6

Universal Messaging 9.6 is the successor of Universal Messaging 9.5.

Universal Messaging 9.6 includes new features, enhancements, and changes as described in the following topics.

Improved JNDI Usability

When JNDI connection factories are managed using webMethods Universal Messaging Enterprise Manager, the JMS URL can now be specified during connection factory creation. In addition, the URL of existing connection factories can now be viewed or changed.

webMethods Suite Integration

Universal Messaging support in various webMethods products has been further extended. For details, see the webMethods Suite Release Notes for the various products.

Java Transport API

To provide a broker-less style of messaging, Universal Messaging now contains a new API, com.softwareag.um.io, which exposes the underlying communication drivers used within the Universal Messaging client and server, and enables direct synchronous and asynchronous communication between client applications using TCP/SSL sockets, SHM (shared memory), or RDMA protocols.

SASL Support

Universal Messaging now provides new security authentication features offering password authentication (Plain and Cram-MD5) via SASL. The Universal Messaging server now accepts username and password credentials from the client, and enables administrators to lock down servers in accordance with specific pluggable providers configured on the Universal Messaging server. These pluggable directory providers include User File (similar to .htaccess files) and LDAP.

MQTT Virtual Payload Types

Universal Messaging now provides MQTT users with the ability to define namespace roots where advanced namespace semantics can be applied. Specifically, the last part of the full topic namespace address can be used to publish / subscribe, indicating your preferred virtual payload type.

Added, Removed, Deprecated, or Changed APIs

The following new API is available:

- com.softwareag.um.io:

Provides direct, broker-less synchronous and asynchronous communication between client applications, using Universal Messaging TCP/SSL sockets, SHM (shared memory), or RDMA protocols.

10 What's New In Universal Messaging 9.5

Universal Messaging 9.5 is the successor of Universal Messaging 9.0/9.1.

Universal Messaging 9.5 includes new features, enhancements, and changes as described in the following topics.

Command Central Support

webMethods Universal Messaging can now be managed and monitored using webMethods Command Central. Users can perform actions such as starting or stopping Universal Messaging realms, monitoring status and KPIs, and configuring ports and license keys of Universal Messaging realms. For details, see the webMethods Command Central section in these release notes.

Deployer Support

Universal Messaging assets can now be deployed using webMethods Deployer. Universal Messaging assets can be exported using Enterprise Manager and checked into a version control system. From there, users can create builds and deploy assets selectively using Deployer. Supported asset types are as follows:

- Realm ACLs
- Security groups
- Realm schedules
- Realm configuration
- Channels
- Channel joins
- Queues
- Interfaces (ports)
- Data groups
- Clusters

Optimize for Infrastructure Support

Universal Messaging can now be monitored using webMethods Optimize for Infrastructure. Fifty-two KPIs are available that span realms, queues, channels, and data groups-all of which can be tracked over time for trending and alerting using advanced analytic capabilities in Optimize.

Single Message Acknowledgement and Redelivery Count

Universal Messaging now supports acknowledgement of individual messages and redelivery counts both on shared durable topics and on queues. This support improves performance of multi-threaded processing and duplicate detection, as is typically used with webMethods Integration Server triggers

Multicast Channel Delivery

Universal Messaging now supports multicast delivery on non-filtered channels. This support allows the high-performance fanout of messages to a large number of interested clients, which is accomplished through the use of broadcast network protocols rather than point-to-point delivery to individual subscribers.

Broader C++ Platform Support

Universal Messaging C++ clients now support a wider range of platforms. These platforms include the following:

- Windows 32- and 64-bit
- Linux 32- and 64-bit
- Mac OS

Persistent Storage Performance Improvements

The performance of Universal Messaging persistent document stores has been improved for mixed and reliable channels.

Shared Memory Performance Improvements

The shared-memory transport capability of Universal Messaging -for low-latency messaging within a single machine - has been improved, further reducing end-to-end latency.

Off Heap Stores

Universal Messaging 9.5 SP2 introduces a new store type for channels and queues - Off Heap. Off Heap Store is a new Topic or Queue store mechanism that uses memory which is not within the Java Heap space, but rather, is allocated directly from the host's memory.

Any memory allocated within the JVM is subject to Garbage Collection inspection. This inspection allows the JVM to release unused memory and move memory that has been used for a while into different memory partitions. It also adds a level of jitter to the JVM, as it needs to pause while it does this inspection and potential move. The use of Off Heap memory stops the Garbage Collection from inspecting and moving these regions since they are outside of the JVM's memory domain. This has the effect of reducing such jitter within the Universal Messaging Server.

Since the events are stored in memory, you still get fast memory access, with no impact from GC within the Server. This is extremely useful when using stores that can

potentially contain data that will exist for a prolonged period of time, since pauses that might otherwise be caused by GC inspections will not occur. By default, the server will be allowed to use a maximum of 1GB of memory for off heap store use (in addition to the current default of 1GB maximum available to the Java heap). However, these amounts are not pre-located. Typically, a system will start off consuming approximately 128-256MB of RAM, and will consume more memory only if it needs to.

RDMA

Remote Direct Memory Access is direct access from the memory of one computer into that of another without involving either one's operating system. This permits high-throughput low-latency communication, which is especially useful in massively parallel computer clusters.

With Shared Memory drivers built into Universal Messaging and supported in C#, C++ and Java we can achieve very fast, high throughput messaging on the same machine. With conventional networking standards like TCP/IP, all messages are pushed through the kernel's network stack and back again. There are some off-loaded drivers allowing application code direct access to the Network card, but these still require a layer of TCP above to interpret and manage. RDMA removes this restriction and allows an area of application memory to be mapped and then written to from a remote computer with no Operating System involvement at all; instead, the Interface card and driver alone marshal the incoming event and put the data directly in the application's memory. This results in performance close to that of same-machine Shared Memory, but between 2 separate computers.

Changes to Default Configurations

The default value for *Config* / *Event Storage* / *CacheAge* has been reduced from 1 day to 1 minute.

The default value for: *Config* / *Global Values* / *DisableExplicitGC* has been changed from *true* to *false*.

A new configuration parameter: *Config* / *Fanout Values* / *DelayPublishOnCapacity* has been added, with a default value of *true*. It causes the publisher to be throttled once any optionally set capacities are reached on a channel or queue.

The default heap size has been changed from 512MB to 1GB.

Miscellaneous API updates

The following updates are available at the API level:

- nConsumeEvent class has new method:

  ack(boolean isSynchronous, boolean ackPrevious)

  This sends an ack for a specific event to the server.

- nConsumeEvent class has new method:

  rollback(boolean isSynchronous)

This sends a rollback for this event to the server.

- nQueueTransactionReader class has new method:

  commit(long eventId)

  This commits all events up to the event ID specified. This means you can partially commit received events.

- nQueueTransactionReader class has new method:

  rollback(long eventId, boolean isIndividual)

  If isIndividual is true, then just the event with the specified event ID is rolled back. This event is then pushed back onto the queue for redelivery. If isIndividual is false, then all events consumed after the specified event ID are rolled back.

- nStoreProperties class has new method:

  enableMulticast(boolean flag)

  If flag is true, then the channel/topic will be multicast-enabled. If false, it won't (default).

11 What's New In Universal Messaging 9.0/9.1

Universal Messaging 9.0/9.1 is the successor of webMethods Nirvana Messaging 7.0.

Universal Messaging 9.0/9.1 includes new features, enhancements, and changes as described in the following topics.

Priority Messaging

Universal Messaging includes support for priority messaging, where higher priority messages are processed before lower priority messages in the client and server queues. For example, if a client queue contains five messages with priorities 0-4, and another message with priority 9 is added to the queue, the next message pulled from the queue is the message with priority 9. This functionality is also used within the client APIs, such that the client APIs also attempt to deliver messages to the client applications based on their priority.

Shared Durable Subscription

Universal Messaging introduces the concept of shared durable subscriptions. Durable subscriptions are topic based, which means all events on a topic are delivered to all consumers; however, only one active consumer is typically permitted to bind itself to a durable subscription name. Multiple active subscribers, such as a cluster of Integration Servers, can now bind to a subscription name, and each message is delivered to only one of these consumers.

Security Groups

Security groups have been introduced in Universal Messaging. Security groups provide the ability to create groups of users and then assign permissions to those groups for Universal Messaging resources, such as realm ACLs and channel or queue ACLs. Using security groups reduces resource usage and simplifies the setup required by applications that use Universal Messaging to create their user permissions.

For example, if a user requires access to the realm and 100 channels, the user must typically add a realm ACL entry and 100 separate ACL entries, one per channel, using the administration API. Using security groups, applications can instead add a security group or groups to the realm API, as well as the 100 channels, and simply make one administration API call to add a user to the security group. Membership of the security groups then implies permission on all resources authorized for the security group.

Installation Using Software AG Installer

Universal Messaging can be installed using the Software AG Installer, which greatly simplifies the installation and initial configuration of Universal Messaging in combination with other webMethods products. This enhancement brings Universal Messaging in line with the proven Software AG installation framework.

Software AG License Key

Universal Messaging uses the standard Software AG license key for authorizing the use of the product. A license key can be provided during installation or added later. If a license key is not provided during installation, a trial license key is installed that allows unrestricted use of Universal Messaging for 90 days. Existing Universal Messaging customers can request a license key by email from keymaster@softwareag.com.