# webMethods Universal Messaging 9.12 Readme

# October 2016

This file contains important information you must read before using webMethods Universal Messaging 9.12. You can find system requirements, user documentation, and installation and upgrade instructions on the Documentation website or the TECHcommunity website. At those locations, you can also find suite-related security and globalization information.

Included in this file is information about functionality that has been added, removed, deprecated, or changed for this product. Deprecated functionality continues to work and is supported by Software AG, but may be removed in a future release. Software AG recommends against using deprecated functionality in new projects.

# 1.0 Critical Information

This section lists any critical issues for the current release that were known when this readme was published. For critical information found later, go to the Knowledge Center on the Empower website.

# 2.0 Known Issues

- Shared Durable Filter displayed in Enterprise Manager

  In the durable objects panel in Enterprise Manager, the displayed shared durable filter does not update dynamically. It will show the correct filter from when the durables are initially loaded, but the displayed filter will not reflect subsequent changes until Enterprise Manager is restarted.

- NUM-6147
  P2P hidden channel appearing

  When upgrading an installation from 9.10 to 9.12, a channel with the name "-p2p/serviceinfo" may be visible within the namespace of Universal Messaging. This channel can safely be removed as it is an artifact of a deprecated and now removed feature P2P. If not removed it will have no adverse effect on Universal Messaging.

- Shared memory drivers are currently not supported on HP-UX systems

  On HP-UX systems, shared memory drivers are currently not supported.
  There is currently no workaround for this issue.

- NUM-2211
  Paged Store type is unsupported in Solaris / SPARC with JDK 1.8.

  Currently, the Paged Storage feature is not supported in a Solaris/SPARC environment with JDK 1.8.

  Workaround: Consider using JDK 1.7 instead.

- PIF-12248
  Cannot deploy JNDI assets from a source Universal Messaging server to a target Universal Messaging server using webMethods Deployer.

  To resolve this issue, install the latest Universal Messaging fixes.

# 3.0 Usage Notes

This section provides any additional information you need to work with the current release of this product.

# 4.0 Fixes Included in Each Release

This section lists the latest fix level that has been included in each release for each product component. Information for a release is listed in this section only if changes occurred in that release. Go to the Knowledge Center on the Empower website for detailed information about fixes.

## *Release 9.12*

## *Release 9.10*

- NUM_9.10.0_Client_Fix4
- NUM_9.10.0_EnterpriseManager_Fix4
- NUM_9.10.0_RealmServer_Fix4
- NUM_9.10.0_TemplateApplications_Fix4

## *Release 9.9*

- NUM_9.9.0_Client_Fix9
- NUM_9.9.0_EnterpriseManager_Fix9
- NUM_9.9.0_wM_RealmServer_Fix9
- NUM_9.9.0_wM_TemplateApplications_Fix9

## *Release 9.8*

- NUM_9.8.0_Client_Fix14
- NUM_9.8.0_RealmServer_Fix14
- NUM_9.8.0_EnterpriseManager_Fix13
- NUM_9.8.0_TemplateApplications_Fix14
- NUM_9.8.0_PlatformManagerPlugins_Fix4

## *Release 9.7*

- NUM_9.7.0_Client_Fix17
- NUM_9.7.0_RealmServer_Fix17
- NUM_9.7.0_EnterpriseManager_Fix14
- NUM_9.7.0_TemplateApplications_Fix17
- NUM_9.7.0_PlatformManagerPlugins_Fix4

## *Release 9.6*

- NUM_9.6.0_Client_Fix18
- NUM_9.6.0_RealmServer_Fix18
- NUM_9.6.0_EnterpriseManager_Fix12
- NUM_9.6.0_TemplateAplications_Fix10

## *Release 9.5 SP2*

- NUM_9.5.2_Client_Fix16
- NUM_9.5.2_RealmServer_Fix16
- NUM_9.5.2_EnterpriseManager_Fix15
- NUM_9.5.2_TemplateApplications_Fix15

## *Release 9.0*

- NUM_9.0.1_Client_Fix14
- NUM_9.0.1_RealmServer_Fix13
- NUM_9.0.1_EnterpriseManager_Fix12

Enhancements:

- 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.

# 5.0 Other Resolved Issues

This section lists the issues that were resolved in each release but were not part of the fixes listed in the previous section. Information for a release is listed in this section only if changes occurred in that release.

## *Release 9.12*

## *Release 9.10*

## *Release 9.9*

- NUM-2769
  Universal Messaging / JMS Temp Queue / Topics must be destroyed on disconnect
  Temporary queues are a JMS feature. When a client is closed, the temporary queues should be deleted. In some cases, for example if a client connection dropped due to a network problem, this was not happening correctly. Now the temporary queues will always be deleted correctly.
  The issue is resolved.

## *Release 9.8*

- NUM-2614
  Universal Messaging Client with semicolon-separated RNAME list always connects to the first entry when there are two nodes in the list.
  The issue is resolved.
- NUM-2427
  Join information not working correctly with .net admin API.
  nChannel.getJoinInformation is returning blank results instead of the correct list of joins fetched from the server.
  The issue is resolved.

## *Release 9.7*

- NUM-1396
  Universal Messaging does not provide transparent failover in cluster
  Example: there is a cluster with two nodes (two sites, second site with isPrime=true). If we switch the node which is not in the isPrime-site off, we would expect a transparent failover. Instead we see the following behavior in a standalone Java client and in a Message-Driven Bean running in JBoss 4.2.3 (using the Universal Messaging 9.5 client libs): Standalone client: for about 10 seconds sends fail with a "nSessionNotConnectedException". If these are caught and the sends are retried, the client can recover. JBoss MDB: stops receiving messages and never does a failover. Even if the node is brought up again, the MDB does not recover.
  This issue is now resolved. The value of the nirvana.conxExceptionOnRetryFailure property of a JMS connection factory determines whether the Universal Messaging cluster fails over

transparently when the prime site fails or throws an exception to the JMS client. For transparent cluster failover, set this property to true. By default, the value of the nirvana.conxExceptionOnRetryFailure property is false.

# 6.0 Documentation Changes

This section describes significant changes to the documentation, such as the addition, relocation, or removal of product guides, online help, chapters, or other major content. A release is listed in this section only if changes occurred in that release.

## *Release 9.12*

The description of how to use Command Central to perform administrative tasks on Universal Messaging has moved from the Command Central documentation set to the Administration Guide of the Universal Messaging documentation set.

## *Release 9.7*

The documentation is provided in a new format, with a navigation frame and content frame. The navigation presents the documentation under the following main categories:

- Concepts
- Administration
- Developer's Guide
- Reference Guide
- Installation Guide

The API documentation is available as a topic in the Reference Guide. The code examples for each of the supported languages are available within the appropriate language-specific part of the Developer's Guide.

# 7.0 Terminology Changes

Information for a release is listed in this section only if changes occurred in that release.

## *Release 9.7*

| Old Term | New Term |
|---|---|
| Brokerless API | umTransport API |

# 8.0 Added, Removed, Deprecated, or Changed Items

This section lists functionality, controls, portlets, properties, or other items that have been added, removed, deprecated, or changed. Information for a release is listed in this section only if changes occurred in that release.

## *Release 9.12*

| Added Item | Description |
|---|---|
| Origin Header to HTTP Drivers | The Universal Messaging WebSocket/HTTP drivers have been updated to process the Origin header field (RFC-6454, RFC-6455 and W3C Cross Origin Resource Sharing).<br><br>Any nhp/nhps interfaces in Universal Messaging 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. |
| Server-side throttling | The server tracks memory utilization.<br><br>When memory utilization is too high, the server will apply progressive levels of back pressure on publishers to prevent an out-of-memory (OOM) event. |
| Improved Logging | Universal Messaging logging has been improved to optionally support Log4J 2 and Logback engines. |
| ACL setting in client API at store creation time | ACLs can now be set at store-creation time via the Client API. This allows basic ACL control for newly created stores without needing the administration API. |

## *Release 9.10*

| Removed Item | Description |
|---|---|
| Standalone Installer | There is no longer a standalone installer for the product. Installation is now available only using the Software AG Installer. |

| Added Item | Description |
|---|---|
| Docker Support | A Universal Messaging Packaging Kit for Docker is now part of the standard Universal Messaging |

| Added Item | Description |
| --- | --- |
| | installation on Linux.<br><br>The Universal Messaging Packaging Kit can be found here: <SAG Install Folder>/UniversalMessaging/server/< Universal Messaging Server Name>/bin/docker.<br><br>The kit includes following Docker tools:<br><br>• Dockerfile for creating Docker image from the Universal Messaging installation on Linux.<br><br>• Samples showing how to start the Universal Messaging server from the Docker image and running Universal Messaging's sample applications within the image.<br><br>• The Tradespace demo application showing how to use 'docker-compose' to set up and run Universal Messaging's TradeSpace demo. |
| New Persistent Store backing | We have a new persistent store backing which, when enabled, removes the need to "perform maintenance" on channels and queues. This new mechanism also allows stores to grow without any restriction on the JVM heap. Our existing persistent store mechanism keeps an in-memory index which grows each time a message is added |

| Added Item | Description |
| --- | --- |
| 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 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, they will be redirected to the master node. Connecting to the master node can give better performance in some use cases. |
| Transactions over AMQP are now supported | The AMQP specification defines a good number of transactional operations. With Universal Messaging 9.10 we now support 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.<br><br>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 the 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. |

## *Release 9.9*

| Added Item | Description |
|---|---|
| AMQP 1.0 Support | Universal Messaging 9.9 introduces support for AMQP 1.0. AMQP is an open standard binary application layer protocol whose main topics of concern are orientation, queuing, routing (both point-to-point and publish-and-subscribe), reliability and security. It defines a number of mechanisms such as flow control, message-delivery guarantee strategies (at-most-once, at-least-once and exactly-once), authentication based on SASL or TLS, etc.<br><br>The Universal Messaging AMQP implementation provides support for a subset of the components of the protocol specification (for more information on AMQP, please refer to http://www.amqp.org/). Here is the complete list of features that we support:<br><br>•AMQP PLAIN & AMQP over SASL<br><br>•Publishing to Topics and Queues<br><br>•Temporary Queues<br><br>•Keep Alives<br><br>•Synchronous and Asynchronous consuming on topics and queues<br><br>•Flow control<br><br>•Durable subscribers<br><br>•Interoperability with existing AMQP JMS clients (SwiftMQ, Apache QPID and Apache Legacy QPID) |
| 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. |

| Added Item | Description |
|---|---|
| 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. |
| Enterprise Manager JNDI panel improvements | The JNDI panel has been reworked to make it more user friendly. It now automatically shows the contents for the server you are connected to and does not require entry of a server URL. It is also clearer how to enable shared durables on creation of a connection factory. |

## *Release 9.8*

| Added Item | Description |
|---|---|
| WAN interest propagation (Realm Zones) | Zones provide a logical grouping of one or more Realms which maintain active connections to each other. Using zones, Universal Messaging can deliver messages between realms and clusters when interest from one realm or cluster has been registered on another realm or cluster in the same zone. Interest between zones is propagated by manually creating static joins. |

| Added Item | Description |
| --- | --- |
| 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. |
| JNDI assets: Connection Factories, Topic Connection Factories, Queue Connection Factories, XA Connection Factories, JNDI Topics, JNDI Queues. | JNDI assets from a source Universal Messaging server can be deployed to a target Universal Messaging server using webMethods Deployer. |

| Removed Item | Replacement, if any |
| --- | --- |
| MQTT Virtual Payload types | Virtual payload type support has now been removed for reasons of compliance to the specification. All MQTT messages are now being handled as JMS bytes messages. |

## *Release 9.7*

| Added Item | Description |
| --- | --- |
| Interest Propagation | Interest Propagation expands upon the functionality provided by joins to provide the ability to forward information only when there are subscribers present on the remote realm or cluster. By forwarding only events which have an active subscription present reduces the number of events and bandwidth used on these links. |
| umTransport API (formerly Brokerless API) support for C++ | The umTransport API was already available for Java, and now this API is available for C++ also, with some restrictions in functionality compared to the Java-based API. |

## *Release 9.6*

| Added Item | Description |
|---|---|
| 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. |

## *Release 9.5 SP1*

| Added Item | Description |
|---|---|
| Multicast for Channel/Topic Delivery | Channels/topics now support delivery of events via multicast. Previously, multicast support was limited to Data Group and inter-cluster communication. As multicast is a broadcast mechanism, the consumers can receive events through Multicast only when they do not use selectors in their subscriptions to channels. |

| Added Item | Description |
| --- | --- |
| Individual Message Acknowledgement | With both topic and queue delivery in JMS, the acknowledge model means that if you consume messages 1 through 10, but acknowledge message 5 only, then messages 1 to 4 are also automatically acknowledged. While this is fine in most cases, there is sometimes the requirement for applications to distribute the consumed messages across different threads or other components, which makes this model a difficult fit.<br><br>We have therefore provided support for acknowledging individual messages, which ensures that only the specified message will be acknowledged. |

## *Release 9.5 SP2*

| Added Item | Description |
| --- | --- |
| Multicast for Channel/Topic Delivery | Channels/topics now support delivery of events via multicast. Previously, multicast support was limited to Data Group and inter-cluster communication. As multicast is a broadcast mechanism, the consumers can receive events through Multicast only when they do not use selectors in their subscriptions to channels. |
| Individual Message Acknowledgement | With both topic and queue delivery in JMS, the acknowledge model means that if you consume messages 1 through 10, but acknowledge message 5 only, then messages 1 to 4 are also automatically acknowledged. While this is fine in most cases, there is sometimes the requirement for applications to distribute the consumed messages across different threads or other components, which makes this model a difficult fit.<br>We have therefore provided support for acknowledging individual messages, which ensures that only the specified message will be acknowledged. |

# 9.0 Added, Removed, Deprecated, or Changed APIs

Information for a release is listed in this section only if changes occurred in that release.

## *Release 9.12*

| Removed Item | Description |
|---|---|
| API for P2P | The API for P2P was deprecated in 9.10 and has been removed for 9.12. |
| Support for Adobe Flex | Support for Adobe Flex was deprecated in 9.10 and has now been removed in 9.12. |

## *Release 9.9*

| Added API | Description |
|---|---|
| Storage API | 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. |

## *Release 9.8*

| Added API | Description |
|---|---|
| com.pcbsys.nirvana.nAdminAPI.nRealmNode<br>▪ public Zone createZone(String zoneName)<br>▪ public void joinZone(Zone zone)<br>▪ public void leaveZone(Zone zone)<br>▪ public Zone getZone() | 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. |

## Release 9.6

| Added API | Description |
|---|---|
| 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. |

## Release 9.5 SP1

| Added API | Description |
|---|---|
| **nConsumeEvent** class has new method:<br>*ack(boolean isSynchronous, boolean ackPrevious)* | Sends an ack for a specific event to the server. |
| **nConsumeEvent** class has new method:<br>*rollback(boolean isSynchronous)* | Sends a rollback for this event to the server. |
| **nQueueTransactionReader** class has new method:<br>*commit(long eventId)* | Commits all events up to the event ID specified. This means you can partially commit received events. |
| **nQueueTransactionReader** class has new method:<br>*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:<br>*enableMulticast(boolean flag)* | If *flag* is true, then the channel/topic will be multicast-enabled. If false, it won't (default). |

## Release 9.5 SP2

| Added API | Description |
|---|---|
| **nConsumeEvent** class has new method:<br>*ack(boolean isSynchronous, boolean ackPrevious)* | Sends an ack for a specific event to the server. |
| **nConsumeEvent** class has new method:<br>*rollback(boolean isSynchronous)* | Sends a rollback for this event to the server. |
| **nQueueTransactionReader** class has new method:<br>*commit(long eventId)* | Commits all events up to the event ID specified. This means you can partially commit received events. |

| Added API | Description |
| --- | --- |
| **nQueueTransactionReader** class has new method:<br>*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:<br>*enableMulticast(boolean flag)* | If *flag* is true, then the channel/topic will be multicast-enabled. If false, it won't (default). |

# 10.0 Copyright Information



# 11.0 Support

Visit the Empower website to learn about support policies and critical alerts, read technical articles and papers, download products and fixes, submit feature/enhancement requests, and more.

Visit the TECHcommunity website to access additional articles, demos, and tutorials, technical information, samples, useful resources, online discussion forums, and more.

UM-RM-912-20161018