



Red Hat AMQ 7.2

AMQ Clients 2.3 Release Notes

Release Notes for Red Hat AMQ Clients

Red Hat AMQ 7.2 AMQ Clients 2.3 Release Notes

Release Notes for Red Hat AMQ Clients

Legal Notice

Copyright © 2019 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Clients 2.3 release.

Table of Contents

CHAPTER 1. FEATURES	3
CHAPTER 2. ENHANCEMENTS	4
2.1. AMQ SPRING BOOT STARTER	4
CHAPTER 3. RESOLVED ISSUES	5
3.1. AMQ C++	5
CHAPTER 4. KNOWN ISSUES	6
4.1. AMQ PYTHON	6
4.2. AMQ .NET	6
CHAPTER 5. IMPORTANT NOTES	7
5.1. AMQ C++	7
5.2. PREFERRED CLIENTS	7
5.3. LEGACY CLIENTS	8
5.4. UPSTREAM VERSIONS	8
CHAPTER 6. IMPORTANT LINKS	9

CHAPTER 1. FEATURES

- AMQ Clients 2.3 adds support for Microsoft Windows 10 Pro and Windows Server 2016.
- AMQ C++ and AMQ Python can now use a configuration file to control connection behavior.
- This release improves the overall performance of AMQ JMS.

CHAPTER 2. ENHANCEMENTS

2.1. AMQ SPRING BOOT STARTER

- [ENTMQCL-1172](#) - Spring Boot 2.1.x support

AMQ Spring Boot Starter now supports version 2.1 of the Spring Boot framework.

CHAPTER 3. RESOLVED ISSUES

3.1. AMQ C++

- **ENTMQCL-1176 - Drain gets stuck on**

In earlier releases of the product, in some cases the client library failed to disable drain mode when the drain operation was complete.

In this release, the library stops draining credit when the drain operation ends.

CHAPTER 4. KNOWN ISSUES

4.1. AMQ PYTHON

- **ENTMQCL-483 - Selectors fail with non-Unicode strings**

The `Selector` option on `Container.create_receiver()` accepts a string. If the string is not supplied as Unicode (in Python 2, `u"somestring"`), any elements escaped with backslashes might not be processed correctly.

Workaround: Users of Python 2 should use an explicit Unicode string in filter declarations to avoid the problem.

- **ENTMQCL-546 - Transactions introduce unexpected link events**

Starting a transaction internally opens a sending link for controlling the transaction. This special link can trigger extra application events.

Workaround: Code using transactions should ensure link handler functions are processing the link they expect.

4.2. AMQ .NET

- **ENTMQCL-794 - Transactions do not work with .NET Core**

Rolling back transactions when using AMQ .NET on .NET Core is not working as expected.

Workaround: Use .NET Framework 4.5 instead of .NET Core if you require transactions.

CHAPTER 5. IMPORTANT NOTES

5.1. AMQ C++

- **Unsettled interfaces**

The AMQ C++ messaging API includes classes and methods that are not yet proven and can change in future releases. Be aware that use of these interfaces might require changes to your application code in the future.

These interfaces are marked **Unsettled API** in the API reference. They include the interfaces in the `proton::codec` and `proton::io` namespaces and the following interfaces in the `proton` namespace.

- `listen_handler`
- `reconnect_options`
- `ssl_certificate`, `ssl_client_options`, and `ssl_server_options`
- `work_queue` and `work`
- The `on_connection_wake` method on `messaging_handler`
- The `wake` method on `connection`
- The `on_sender_drain_start` and `on_sender_drain_finish` methods on `messaging_handler`
- The `draining` and `return_credit` methods on `sender`
- The `draining` and `drain` methods on `receiver`

API elements present in header files but not yet documented are considered unsettled and are subject to change.

- **Deprecated interfaces**

Interfaces marked **Deprecated** in the API reference are scheduled for removal in a future release.

This release deprecates the following interfaces in the `proton` namespace.

- `void_function0` - Use the `work` class or C++11 lambdas instead.
- `default_container` - Use the `container` class instead.
- `url` and `url_error` - Use a third-party URL library instead.

5.2. PREFERRED CLIENTS

In general, AMQ clients that support the AMQP 1.0 standard are preferred for new application development. However, the following exceptions apply:

- If your implementation requires distributed transactions, use the AMQ Core Protocol JMS client.

- If you require MQTT or STOMP in your domain (for IoT applications, for instance), use community-supported MQTT or STOMP clients.

The considerations above do not necessarily apply if you are already using:

- The AMQ OpenWire JMS client (the JMS implementation previously provided in A-MQ 6)
- The AMQ Core Protocol JMS client (the JMS implementation previously provided with HornetQ)

5.3. LEGACY CLIENTS

- **Deprecation of the CMS and NMS APIs**

The ActiveMQ CMS and NMS messaging APIs are deprecated in AMQ 7. It is recommended that users of the CMS API migrate to AMQ C++, and users of the NMS API migrate to AMQ .NET. The CMS and NMS APIs might have reduced functionality in AMQ 7.

- **Deprecation of the legacy AMQ C++ client**

The legacy AMQ C++ client (the C++ client previously provided in MRG Messaging) is deprecated in AMQ 7. It is recommended that users of this API migrate to AMQ C++.

- **The Core API is unsupported**

The Artemis Core API client is not supported. This client is distinct from the AMQ Core Protocol JMS client, which is supported.

5.4. UPSTREAM VERSIONS

- AMQ C++, AMQ Python, and AMQ Ruby are now based on [Qpid Proton 0.27.0](#)
- AMQ JavaScript is now based on [Rhea 0.3.9](#)
- AMQ JMS is now based on [Qpid JMS 0.40.0](#)
- AMQ .NET is now based on [AMQP.Net Lite 2.1.6](#)

CHAPTER 6. IMPORTANT LINKS

- [Red Hat AMQ 7 Supported Configurations](#)
- [Red Hat AMQ 7 Component Details](#)
- [AMQ Clients 2.2 Release Notes](#)
- [AMQ Clients 2.1 Release Notes](#)
- [AMQ Clients 2.0 Release Notes](#)
- [AMQ Clients 1.2 Release Notes](#)
- [AMQ Clients 1.1 Release Notes](#)

Revised on 2019-03-18 15:33:45 UTC