Red Hat AMQ 7.2

AMQ Clients 2.2 Release Notes

Release Notes for Red Hat AMQ Clients

Last Updated: 2018-11-15

Red Hat AMQ 7.2 AMQ Clients 2.2 Release Notes

Release Notes for Red Hat AMQ Clients

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Clients 2.2 release.

Table of Contents

CHAPTER 1. FEATURES

- A new library for using AMQ with the Spring framework, AMQ Spring Boot Starter, is now available.
- AMQ JMS can now optionally use native OpenSSL libraries for improved SSL/TLS performance.
- This release improves the overall performance of AMQ JMS.

CHAPTER 2. ENHANCEMENTS

2.1. AMQ C++

- **ENTMQCL-727 - Add a thread-safe wake operation**
  The client library now provides a `wake()` method on `connection` and a new `on_connection_wake()` event. These serve as thread-handling primitives for use in more advanced multithreaded programs.

CHAPTER 3. RESOLVED ISSUES

3.1. AMQ C++, AMQ PYTHON, AMQ RUBY

- ENTMQCL-943 - **Normalize encoding of AMQP "multiple" fields**
  In earlier releases of the product, the encoding of AMQP fields such as source and target capabilities was incompatible with some AMQP implementations.

  In this release, the encoding is normalized for improved interoperability.

3.2. AMQ C++

- ENTMQCL-693 - **Cannot receive message properties with null values**
  In earlier releases of the product, the client library threw a `conversion_error` if a null value was decoded into a `proton::scalar`. This occurred when decoding application properties that contained null values.

  In this release, the library accepts null values in message properties.

3.3. AMQ JMS

- ENTMQCL-923 - **Stop further reconnection attempts after an unrecoverable authentication error**
  In earlier releases of the product, a client library configured to reconnect continued to attempt new connections after unrecoverable SASL errors such as invalid user credentials.

  In this release, the library stops the reconnection process after an unrecoverable authentication error.

3.4. AMQ .NET

- ENTMQCL-903 - **Incorrect default message priority**
  In earlier releases of the product, the client library reported the priority of a message as 0 if it was not explicitly set. The AMQP specification requires the default be 4.

  In this release, the library returns the correct default priority.

- ENTMQCL-874 - **Cannot reuse TCP socket on Unix**
  In previous releases of the product, the client library failed when attempting to reuse a TCP socket after it was recently closed.

  In this release, the library can reuse recently closed sockets without error.

CHAPTER 4. KNOWN ISSUES

4.1. AMQ PYTHON

- **ENTMQCL-483 - Selectors with backslashes are invalid in 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.
- **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 0.26.0
- AMQ JavaScript is now based on Rhea 0.3.1
- AMQ JMS is now based on Qpid JMS 0.37.0
- AMQ .NET is now based on AMQP.Net Lite 2.1.4

CHAPTER 6. IMPORTANT LINKS

- Red Hat AMQ 7 Supported Configurations
- Red Hat AMQ 7 Component Details
- 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 2018-11-15 13:03:17 UTC