Red Hat JBoss AMQ 7.1

AMQ Clients 1.2 Release Notes

Release Notes for Red Hat JBoss AMQ Clients

Last Updated: 2018-03-09

Red Hat JBoss AMQ 7.1 AMQ Clients 1.2 Release Notes

Release Notes for Red Hat JBoss AMQ Clients

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Clients 1.2 release.

Table of Contents

CHAPTER 1. FEATURES

- AMQ C++ now offers automatic reconnect with connection failover.
- AMQ C++ now supports multithreaded operation on Red Hat Enterprise Linux.
- AMQ C++, AMQ JMS, and AMQ Python now support Kerberos authentication.

CHAPTER 2. ENHANCEMENTS

2.1. AMQ C++

- **ENTMQCL-420 - Improve the API for handling map data**
  The API for map-structured data now provides methods to convert to standard C++ data structures as well as to raw `proton::value` objects.

- **ENTMQCL-516 - Expose connection driver "tick"**
  The `connection_driver` interface now includes a method to trigger additional IO and event processing. This is useful for custom IO integrations.

- **ENTMQCL-527 - Improve the API for scheduling work**
  This release introduces the `work_queue` and `work` interfaces. They enable you to define routines that can be injected into the event-processing stream across thread boundaries or scheduled for later execution.

- **AMQ C++ is now based on Qpid Proton 0.18.0**

2.2. AMQ JAVASCRIPT

- **AMQ JavaScript is now based on Rhea 0.2.5**

2.3. AMQ JMS

- **ENTMQCL-543 - Add ability to reject messages in `JmsRedeliveryPolicy`**
  You can now set the message-acknowledgment policy for redelivered messages using the `jms.redeliveryPolicy.outcome` connection URI option.

- **Add support for the Netty epoll transport**
  If the OS supports it, the client now uses the epoll IO interface for improved performance.

- **AMQ JMS is now based on Qpid JMS 0.26.0**

2.4. AMQ PYTHON

- **AMQ Python is now based on Qpid Proton 0.18.0**

CHAPTER 3. RESOLVED ISSUES

3.1. AMQ JAVASCRIPT

- **ENTMQCL-485 - Filter module unavailable in the browser**
  In earlier releases of the product, the filter module was unavailable in a browser. As a consequence, filters could not be used in browser code.

  In this release, the filter module is made available from the container object and is now usable in browser code.

- **ENTMQCL-490 - Logging prints `[object]` instead of content of object**
  In earlier releases of the product, the code to print object information was not showing component details of the object. As a consequence, log messages had less detail than expected.

  In this release, the object printing code now prints the members representing the object and the log messages have the expected level of detail.

- **ENTMQCL-505 - Examples fail on Node.js 0.10**
  In earlier releases of the product, the example code had a dependency on an arg parsing module that is not available for Node.js 0.10. As a consequence, the examples failed to run.

  In this release, the examples are rewritten to use another arg parsing module, and now the examples can run on Node.js 0.10.

- **ENTMQCL-532 - The browserify command ignores locally installed files**
  In previous releases of the product, the **`browserify`** script did not include locally installed files in its path. As a result, you could not run the script without NPM (node package manager) installing all of your code.

  In the this release, the current directory is now included in the Node.js path and the **`browserify`** script uses code from modules in the current directory.

3.2. AMQ JMS

- **ENTMQCL-511 - Memory leak after remote connection close**
  In earlier releases of the product, various concurrent AMQP operations were triggering memory leaks. As a consequence, client memory use was growing.

  In this release, the leaks are fixed and client memory use is now stable.

3.3. AMQ PYTHON

- **ENTMQCL-470 - Faulty error handling when listener address is already in use**
  In earlier releases of the product, there was an incorrect string format in an error handler. As a consequence, the **`address already in use`** error message was obscured.

  In this release, the string format is corrected and the error message is no longer obscured.

- **ENTMQCL-471 - Creating a container leaks two file descriptors**
  In earlier releases of the product, the client was not cleaning up all references to internal objects. As a consequence, file descriptors were not being freed after container deletion.

  In this release, the client frees all of its file descriptors on container deletion.

CHAPTER 4. KNOWN ISSUES

4.1. AMQ C++

- **ENTMQCL-565 - Windows build does not compile the examples that require C++11**
  The AMQ C++ client is supported for use on Microsoft Windows Server 2012 R2. However, compiling the examples as C++11 is not yet supported on this operating system.

- **ENTMQCL-584 - Kerberos authentication cannot proceed if the user option is not set**
  In order for Kerberos authentication to succeed, a value must be supplied to the `user` connection option. This value need not match the credential established using `kinit`.

- **ENTMQCL-604 - Receiver name is not set when using the Container.create_receiver option**
  Due to a flaw in the `Container.create_receiver` method processing, the receiver name is not set when using the `Container.create_receiver` option.

  Workaround: Set the receiver name using the`Connection.create_receiver` option instead of the `Container.create_receiver` option.

4.2. 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.

4.3. AMQ .NET

- **ENTMQCL-500 - Receive does not raise exception when link closed with error**
  Pipelined protocol state events can prevent the client from raising link errors to the API user. Notification of errors might occur in subsequent event handling functions.

4.4. AMQ JMS

- **ENTMQCL-606 - Interconnect fails to authenticate using Kerberos**
  When Interconnect tries to authenticate using Kerberos, an unexpected empty sasl-challenge is sent to the client. Due to the empty sasl-challenge, the connection establishment process fails to complete and times out.

CHAPTER 5. IMPORTANT NOTES

5.1. 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 (the JMS implementation previously provided with HornetQ).
- 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)

5.2. NMS AND CMS

- **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.

5.3. 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.

CHAPTER 6. IMPORTANT LINKS

- Red Hat JBoss AMQ 7 Supported Configurations
- Red Hat JBoss AMQ 7 Component Details
- AMQ Clients 1.1 Release Notes

Revised on 2018-03-09 16:38:07 EST