Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Clients 2.5 release.
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHAPTER 1. FEATURES</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>CHAPTER 2. ENHANCEMENTS</strong></td>
<td>4</td>
</tr>
<tr>
<td>2.1. AMQ C++, AMQ PYTHON, AMQ RUBY</td>
<td>4</td>
</tr>
<tr>
<td>2.2. AMQ C++</td>
<td>4</td>
</tr>
<tr>
<td><strong>CHAPTER 3. FIXED ISSUES</strong></td>
<td>5</td>
</tr>
<tr>
<td>3.1. AMQ C++, AMQ PYTHON, AMQ RUBY</td>
<td>5</td>
</tr>
<tr>
<td><strong>CHAPTER 4. KNOWN ISSUES</strong></td>
<td>6</td>
</tr>
<tr>
<td><strong>CHAPTER 5. IMPORTANT NOTES</strong></td>
<td>7</td>
</tr>
<tr>
<td>5.1. AMQ C++</td>
<td>7</td>
</tr>
<tr>
<td>5.2. PREFERRED CLIENTS</td>
<td>7</td>
</tr>
<tr>
<td>5.3. LEGACY CLIENTS</td>
<td>8</td>
</tr>
<tr>
<td>5.4. UPSTREAM VERSIONS</td>
<td>8</td>
</tr>
<tr>
<td><strong>CHAPTER 6. IMPORTANT LINKS</strong></td>
<td>9</td>
</tr>
</tbody>
</table>
CHAPTER 1. FEATURES

- AMQ .NET now supports .NET Core 2.1.
- AMQ JavaScript now supports file-based configuration.
- File-based configuration is now available on Windows.
- AMQ JMS and AMQ Python now offer distributed tracing based on OpenTracing.

IMPORTANT

Distributed tracing in AMQ Clients is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.
CHAPTER 2. ENHANCEMENTS

2.1. AMQ C++, AMQ PYTHON, AMQ RUBY

- **ENTMQCL-1358** - Use the system certificate authority by default
  When SSL/TLS is enabled, the clients now use the system-defined certificate authority by default if the user does not specify one.

2.2. AMQ C++

- **ENTMQCL-1282** - Use library-level defaults for no-args connect
  The client `connect()` method with no arguments now uses library-defined default connection options if the `connect.json` file is not found. Previously it raised an exception.

- **ENTMQCL-1577** - Allow connection options to be updated for automatic reconnect
  The client API now offers methods for updating the connection options used during reconnect attempts.
CHAPTER 3. FIXED ISSUES

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

- **ENTMQCL-1376** - The settlement callback is not called when the client receives multiple disposition updates
  
  In earlier releases of the product, the client could fail to call its settlement callback (on_tracker_settle or on_settled) when it received multiple disposition updates, such as when it received released from the remote peer but did not yet receive settled.

  In this release, the client only auto-settles messages after it gets the settlement from the receiver, ensuring that the settlement callback is called at the right time.
CHAPTER 4. KNOWN ISSUES

- **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.
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`
  - 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 0.29.0
- AMQ JavaScript is now based on Rhea 1.0.8
- AMQ JMS is now based on Qpid JMS 0.45.0
- AMQ .NET is now based on AMQP.Net Lite 2.1.8
CHAPTER 6. IMPORTANT LINKS

- Red Hat AMQ Supported Configurations
- Red Hat AMQ 7 Component Details
- AMQ Clients 2.4 Release Notes
- AMQ Clients 2.3 Release Notes
- 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 2019-10-09 14:23:13 UTC