Red Hat AMQ Broker 7.11

Release Notes for Red Hat AMQ Broker 7.11

Release Notes for AMQ Broker

Last Updated: 2024-04-18

Red Hat AMQ Broker 7.11 Release Notes for Red Hat AMQ Broker 7.11

Release Notes for AMQ Broker

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Broker 7.11 release.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. LONG TERM SUPPORT FOR AMQ BROKER 7.11

AMQ Broker 7.11 has been designated as a Long Term Support (LTS) release version. For details of the terms of an LTS release, see How long are AMQ LTS releases supported?

Support for Red Hat Enterprise Linux and OpenShift Container Platform

The AMQ Broker 7.11 LTS version supports:

- Red Hat Enterprise Linux 7, 8 and 9
- OpenShift Container Platform 4.12, 4.13, 4.14 or 4.15

Red Hat strives to ensure that AMQ Broker remains compatible with future versions of OpenShift Container Platform; however this compatibility cannot be guaranteed. Interoperability testing is performed for each new OpenShift Container Platform version. If no compatibility issues are found, the new OpenShift Container Platform version is added to the Red Hat AMQ Broker 7 Supported Configurations.

CHAPTER 2. SUPPORTED CONFIGURATIONS

For information on supported configurations, see Red Hat AMQ Broker 7 Supported Configurations .

Minimum Java version

At a minimum, AMQ Broker 7.11 requires Java version 11 to run.

Openwire support

AMQ 7 Broker has provided support for the Openwire protocol since its release in 2017 as a means to migrate client applications to AMQ 7. With the release of AMQ Broker 7.9.0 in 2021, the Openwire protocol was deprecated and customers were encouraged to migrate their existing Openwire client applications to one of the fully supported protocols of AMQ 7 (CORE, AMQP, MQTT, or STOMP). Starting with the AMQ Broker 7.12.0 release, support for the Openwire protocol will no longer be provided.

CHAPTER 3. NEW AND CHANGED FEATURES

This section describes a highlighted set of enhancements and new features in AMQ Broker 7.11.

Validation changes for image and version attributes in a CR

In 7.11.5, the Operator validates the configuration of image and version attributes in a CR differently to previous versions.
Before 7.11.5, the Operator validates that a CR does not have:

- A **spec.deploymentPlan.image** attribute without a **spec.deploymentPlan.initImage** attribute or vice versa.
- A **spec.version** attribute with either a **spec.deploymentPlan.image** and a **spec.deploymentPlan.initImage** attribute, or both.

In 7.11.5, the Operator continues to validate that a CR does not have a **spec.deploymentPlan.image** attribute without a **spec.deploymentPlan.initImage** attribute or vice versa. In 7.11.5, the Operator validation is changed to also verify that:

- The version number specified in the **spec.version** attribute, if present, matches the version of the broker container images deployed.
  If the versions are not the same, the Operator sets the status of the **BrokerVersionAligned** condition in the CR to **Unknown** to highlight the mismatch for information purposes, but the mismatch does not affect the running of the brokers in the deployment.
- A CR does not have **spec.deploymentPlan.image** and **spec.deploymentPlan.initImage** attributes without a **spec.version** attribute.
  If a CR has **spec.deploymentPlan.image** and **spec.deploymentPlan.initImage** attributes without a **spec.version** attribute, the Operator sets the status of the **Valid** condition in the CR to **Unknown** to warn that the configuration is incomplete.

  NOTE

  A missing **spec.version** attribute causes the Operator to restart the broker Pods in the deployment each time the Operator is upgraded. The Pod restart is required because the Operator updates a label in the StatefulSet with the latest supported broker version, unless a version number is explicitly set in the **spec.version** attribute.

  To prevent each future Operator upgrade from restarting the broker, you must set the version number of the broker deployed in the **spec.version** attribute in the CR.

  - In 7.11.5, you can find the version number of the broker deployed in the **status** section of the CR **after** you start the broker. For more information, see Viewing status information for your broker deployment in *Deploying AMQ Broker on Openshift*.
  - Prior to 7.11.5, you can find the version number in a broker Pod's log file by using the OpenShift Container Platform web console:
    i. Click **Workloads → Pods**.
    ii. Click a broker Pod name.
    iii. Click the **Logs** tab.

The version number is displayed after the **Artemis** banner at the top of the log output.

NOTE

Conditions that have a status value of **Unknown** do not prevent the Operator from completing the broker deployment.

Leader election settings for the Operator can be customized

Starting in 7.11.5, you can customize the settings used by the Operator for leader elections. If you use Operator Hub to install the Operator, you can use the OpenShift Container Platform web console to configure the leader election settings in the Operator subscription after you install the Operator. If you use the OpenShift Container Platform command-line interface to install the Operator, you can configure the leader elections settings in the Operator configuration file, **operator.yaml**, either before or after you install the Operator.
The following is an example of the leader elections settings configured in the **operator.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
...
template
...
spec:
containers:
- args:
- --leader-elect
- --lease-duration=60
- --renew-deadline=40
- --retry-period=5
...
```

The following is an example of the leader elections settings configured in the Operator subscription:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
...
spec:
...
config:
env:
- name: ARGS
value: "--leader-elect --lease-duration=60 --renew-deadline=60 --retry-period=5"
```

leader-elect: Enables the Operator to compete in a leader election with other instances to ensure that no more than one instance operates at a time.

lease-duration: The duration, in seconds, that a non-leader Operator waits before it attempts to acquire the lease that was not renewed by the previous leader. The default is **15**.

renew-deadline: The duration, in seconds, a leader Operator waits between attempts to renew the leader role before it stops leading. The default is **10**.

retry-period: The duration, in seconds, that the Operator waits between attempts to acquire and renew the leader role. The default is **2**.

Individual attributes in an address-settings element can be updated

In 7.11.5, you can use the Jolokia REST interface to JMX to update individual attributes in an **address-settings** element in JSON format. Previously, if you wanted to update an attribute or subset of attributes in an **address-settings** element, it was necessary to also include all the unchanged **address-settings** attributes in the update operation.

Warning level messages changed to error level in the Critical Analyzer

In 7.11.5, the Critical Analyzer assigns an **ERROR** level to messages that were previously assigned a **WARN** level.

New parameters to provide greater control over the flow of paged messages into memory

Before 7.11.3, you control the flow of paged messages into memory by setting limits for the **max-read-page-bytes** and **max-read-page-messages** parameters. When applying these limits, the broker counts both messages in memory that are ready for delivery to consumers and messages that are currently delivering. If consumers are slow to acknowledge messages, the memory or message limit can be used by messages that are currently delivering, which prevents the broker reading new messages into memory. As a result, the broker can be starved of messages.
Starting in 7.11.3 you can set limits for two new parameters to control the flow of paged messages into memory. When applying these limits the broker does not take into account delivering messages.

- **prefetch-page-bytes** Memory, in bytes, that is available to read paged messages into memory per-queue. The default value is 20MB.
- **prefetch-page-messages** Number of paged messages that the broker can read from disk into memory per-queue. The default value is -1, which means that no limit applies.

If consumers are slow to acknowledge messages, you can increase the default limits for the **max-read-page-bytes** and **max-read-page-message** parameters to provide capacity for delivering messages. The default limits for the **prefetch-page-bytes** and the **prefetch-page-messages** parameters then allow the broker to read new messages into memory.

NOTE

If the value of the **max-read-page-bytes** parameter is reached before the value of the **prefetch-page-bytes** parameter, the broker stops reading further paged messages into memory.

AMQ Core Protocol JMS client can fail over to other live brokers in a cluster

Prior to 7.11.2, if an AMQ Core Protocol JMS client loses the connection to a live broker, it can fail over only to a backup broker when both brokers are configured as a live/backup pair for high availability (HA). Starting in 7.11.2, the AMQ Core Protocol JMS client can be configured to fail over to a backup broker in a HA pair or to any other live broker in a cluster.

To enable the client to fail over to any live broker in a cluster, specify the **failoverAttempts** configuration option in the connection URI of the client. For example:

```
ConnectionFactory connectionFactory = new
ActiveMQConnectionFactory(“(tcp://host1:port,tcp://host2:port,tcp://host3:port,tcp://host4:port,tcp://host5
:port)?ha=true&failoverAttempts=2&reconnectAttempts=2”);
```

With the **failoverAttempts** option set to a value of 2, the client attempts to connect to 2 other live brokers in the cluster. The failover attempts occur if the client fails all connection attempts to the live broker in a non-HA configuration and the live and backup broker in a HA configuration.

NOTE

The **reconnnectAttempts** configuration option, shown in the example, is used to fail over only from the live broker to a backup broker if both brokers are configured in a HA pair.

Paging performance improvements when AMQ Broker uses JDBC persistence

Starting in 7.11.2, paging performance is improved when AMQ Broker is configured to use JDBC persistence. As part of the paging improvements, a new parameter, **jdbc-max-page-size-bytes**, limits the page size to 100KB by default when using JDBC. You can customize the default limit. For more information, see Configuring JDBC persistence in *Configuring AMQ Broker*.

Batching of federated messages

If the backlog on your queues exceeds the available capacity of local consumers, any lower priority federation consumer becomes a candidate to receive messages. Eventually, too many messages may move to federated consumers and cause the same scenario to occur on the other cluster, with the result that messages move back and forth between brokers.

Starting in 7.11.2, you can configure federated consumers to pull batches of messages only when the local queue has excess capacity. As a result, federation does not move more messages than federated consumers can process, which avoids a scenario where messages move back and forth between brokers.

To configure federated consumers to pull batches of messages, set the **consumerWindowSize** value to **0** on the connection URI of federated consumers.

```
tcp://<host>:<port>?consumerWindowSize=0
```

With the **consumerWindowSize** value set to **0**, AMQ Broker uses the value of the **defaultConsumerWindowSize** attribute in the address settings for a matching address to determine the batch size of messages moved between brokers. The default value for the **defaultConsumerWindowSize** attribute is **1048576** bytes.

Use this batching mode of operation for bidirectional federation between active brokers.

For more information about federation, see Federating addresses and queues in *Configuring AMQ Broker*.

Broker startup health check

Starting in 7.11.2, you can configure a startup probe to check that the AMQ Broker application within an OpenShift Container Platform container started successfully. To learn how to configure health checks, see Configuring broker health checks in *Deploying AMQ Broker on Openshift*.

Limiting disk space used for paging

If you configure AMQ Broker to page messages, you can limit the disk space used to page incoming messages to prevent the paging operation from using excessive disk space. For more information, see Limiting disk usage during paging for specific addresses in *Configuring AMQ Broker*.

Limiting the memory used for messages read from paging

If you configure AMQ Broker to page messages, you can limit the memory used to store messages that the broker transfers from disk back to memory when clients are ready to consume messages. For more information, see Controlling the flow of paged messages into memory in *Configuring AMQ Broker*.

IMPORTANT

If client applications leave too many messages pending acknowledgment, the broker does not read paged messages until the pending messages are acknowledged, which can cause message starvation on the broker.

For example, if the limit for the transfer of paged messages into memory, which is 20MB by default, is reached, the broker waits for an acknowledgment from a client before it reads any more messages. If, at the same time, clients are waiting to receive sufficient messages before they send an acknowledgment to the broker, which is determined by the batch size used by clients, the broker is starved of messages.

To avoid starvation, either increase the broker limits that control the transfer of paged message into memory or reduce the number of delivering messages. You can reduce the number of delivering messages by ensuring that clients either commit message acknowledgments sooner or use a timeout and commit acknowledgments when no more messages are received from the broker.

You can see the number and size of delivering messages in a queue's **Delivering Count** and **Delivering Bytes** metrics in AMQ Management Console.

Log4j 2 logging support

Starting in 7.11, AMQ Broker uses the Log4j 2 logging utility instead of the JBoss Logging framework to provide message logging. You can customize the Log4j 2 logging configuration on both OpenShift Container Platform and RHEL platforms.

Changing the Operator logging level

In AMQ Broker 7.11 on OpenShift Container Platform, you can change the default logging level to increase or decrease the detail that is written to the Operator logs. For more information, see Changing the logging level for the Operator in *Deploying AMQ Broker on Openshift*.

Support for Java Authentication and Authorization Service (JAAS) login modules

In AMQ Broker 7.11 on OpenShift Container Platform, you can configure JAAS login modules in a secret instead of using the ActiveMQArtemisSecurity CR to configure user authentication and authorization for AMQ Broker. By configuring JAAS login modules in a secret, you can update user and role information in properties files without needing to restart the broker. In addition, you can configure login modules, such as LDAP, which are not configurable in the CR. For more information, see Configuring JAAS login modules in a secret in *Deploying AMQ Broker on Openshift*.

Restricting upgrades

In AMQ Broker 7.11 on OpenShift Container Platform, the Operator automatically upgrades the broker container images to the latest available version. You can configure the Custom Resource (CR) for your deployment to prevent automatic upgrades or to permit automatic upgrades only to specific versions or to specific broker and init container images.

NOTE

If you want to restrict automatic upgrades, you must specify either the combined **spec.deploymentPlan.image** and **spec.deploymentPlan.initImage** attributes or the **spec.version** attribute, but not both, in the CR.

For more information, see Restricting automatic upgrades in *Deploying AMQ Broker on Openshift*.

Extended status reporting

In AMQ Broker 7.11 on OpenShift Container Platform, the status information reported by the Operator in the main broker CR is extended to include:

- The validity of the content of the CR.
- The application of properties configured in the **brokerProperties** attribute.
- The propagation of Java Authentication and Authorization Service (JAAS) login module files in a secret to the broker Pods.
- The version of the broker that is deployed and the URLs of the broker and init container images for that version.
- The ability to apply major, minor, patch and security upgrades to the deployment.

Synchronous mirroring support

Starting in 7.11, you can configure synchronous mirroring between brokers to ensure that messages are written to the volumes of both brokers in the mirror at the same time. By using synchronous mirroring, you ensure that the mirrored broker is up-to-date for disaster recovery. For more information, see Configuring broker connections in *Configuring AMQ Broker*.

Pod disruption budget

In AMQ Broker 7.11 on OpenShift Container Platform, you can configure a Pod disruption budget to specify the minimum number of Pods in a cluster that must be available simultaneously during a voluntary disruption, such as a maintenance window. For more information, see Configuring a Pod disruption budget in *Deploying AMQ Broker on OpenShift*.

brokerProperties configuration is stored in a mutable secret

In AMQ Broker 7.11 on OpenShift Container Platform, the configuration created by using the **brokerProperties** attribute in the CR is stored in a mutable secret. A mutable secret can be updated without requiring a broker restart. Therefore, configuration updates are applied when the broker reloads the configuration periodically, apart from any update that specifically requires a broker restart.

Operations to control the embedded web server

Starting in 7.11, you can stop and restart the embedded web server for AMQ Broker by using the **stopEmbeddedWebServer**, **startEmbeddedWebServer** and **restartEmbeddedWebServer** operations on the ActiveMQServerControl JMX MBean. By using these operations, you can avoid restarting AMQ Broker if, for example, you renew the SSL certificate for AMQ Broker.

The credentials used to log in to AMQ Management Console are used to send messages

In previous versions of AMQ Broker, it was necessary to specify a username and password in the AMQ Management Console **Preferences** page in order to send messages in AMQ Management Console. Starting in 7.11, messages are sent using the credentials that you use to log in to AMQ Management Console.
You can override the default behavior and specify different credentials to send individual messages. For more information, see, Sending messages to an address in *Managing AMQ Broker*.

AMQ Broker on OpenShift Container Platform is preconfigured to collect metric data for Prometheus

In AMQ Broker 7.11 on OpenShift Container Platform, the AMQ Broker container Pods are preconfigured to allow the Prometheus Operator Service Monitor to collect metric data. In previous releases, you needed to expose the port required by Service Monitor to collect metric data.

Setting environment variables for broker containers

In AMQ Broker 7.11 on OpenShift Container Platform, you can set environment variables in a Custom Resource (CR) that are passed to each broker container. For example, you can add the **TZ** environment variable to set the timezone of the broker container. For more information, see Setting environment variables for broker containers in *Deploying AMQ Broker on OpenShift*.

Proxy forwarding support on OpenShift Container Platform

In AMQ Broker 7.11 on OpenShift Container Platform, the embedded web server, which hosts AMQ Management Console, is preconfigured to handle X-Forwarded headers. By handling X-Forwarded headers, AMQ Management Console can receive header information that is otherwise altered or lost when a proxy is involved in the path of a request. For example, AMQ Management Console uses HTTP but the OpenShift Container Platform router, which is a proxy, exposes AMQ Management Console using a HTTPS route that terminates at the router. From the X-Forwarded header, AMQ Management Console can identify that the connection between the browser and the router uses HTTPS and switch to HTTPS to serve browser requests.

Some redelivery attributes are supported only in thebrokerProperties CR attribute

If a 7.8.x or 7.9.x deployment has the **redeliveryDelayMultiplier** or the **redeliveryCollisionAvoidanceFactor** attributes configured in the **spec.deploymentPlan.addressSettings.addressSetting** CR element, you must configure these attributes in the **brokerProperties** CR attribute after you upgrade to 7.11.x. This is required because the data type of both attributes changed from float to string in 7.10.0. As a result, these attributes are no longer supported in the **spec.deploymentPlan.addressSettings.addressSetting** attribute. The following example shows how to configure both attributes in the **brokerProperties** element:

```
spec:
  ...
  brokerProperties:
  - "addressSettings.#.redeliveryMultiplier=2.1"
  - "addressSettings.#.redeliveryCollisionAvoidanceFactor=1.2"
  ...
```

NOTE

In the **brokerProperties** attribute, use the **redeliveryMultiplier** attribute name instead of the **redeliveryDelayMultiplier** attribute name that was used in the **spec.deploymentPlan.addressSettings.addressSetting** element.

Disabling XML External Entity (XXE) processing

If you don't need to modularize your broker configuration in separate files that are included in the **broker.xml** file, you can now disable XXE processing to protect against XXE security vulnerabilities. Where possible, Red Hat recommends that you disable XXE processing. For more information, see Disabling External XML Entity (XXE) processing in *Configuring AMQ Broker*.

JGroups 5.x

Versions of AMQ Broker before 7.10.0 used JGroups 3.x. AMQ Broker 7.11 uses JGroups 5.x which is not backwardly compatible with JGroups 3.x. Some protocols and protocol properties changed between the two JGroup versions, so you may have to change the JGroups stack configuration when you upgrade to AMQ Broker 7.11.

Contents of AMQ Broker archive extracted to a specific directory name

When you extract the AMQ Broker archive on Red Hat Enterprise Linux, the contents of the archive are extracted to a directory called **apache-artemis-2.28.0.redhat-00019** in your current directory.

Operator channels

The AMQ Broker Operator, **Red Hat Integration - AMQ Broker for RHEL 8 (Multiarch)**, is available with the following channels:

- **7.11.x** - This channel provides updates for version 7.11 only and is a Long Term Support (LTS) channel.
- **7.10.x** - This channel provides updates for version 7.10 only and is a Long Term Support (LTS) channel.

NOTE

It is not possible to upgrade the Operator by switching channels. You must uninstall the existing Operator and install the new version of the Operator from the appropriate channel.

To determine which Operator to choose, see the Red Hat Enterprise Linux Container Compatibility Matrix.

Change to class name of the Prometheus metrics plugin

In AMQ Broker 7.11, the class name of the Prometheus metrics plugin that is included with AMQ Broker changed from **org.apache.activemq.artemis.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** to **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin**. If the Prometheus metrics plugin is enabled in a previous version of AMQ Broker, you must update the class name in the **broker.xml** configuration file when you upgrade to AMQ Broker 7.11. For more information, see Upgrading a broker instance from 7.10.x to 7.11.x in *Managing AMQ Broker*.

Changes to the data returned by the listProducers API method and the listProducersInfoAsJSON JMX method

In AMQ Broker 7.11, the following are enhancements to how data is returned by the **listProducers** method, which is used by AMQ Management Console:

- In the data returned in previous versions, a producer was displayed per-session. Therefore, if you created a producer using the Core protocol, which creates two sessions per-producer, separate producers are shown for each of the two sessions. Also, if you created a producer without sending messages from the producer, the address returned for the producer was empty. In AMQ Broker 7.11, the **listProducers method** returns a single producer for the two sessions created by the Core protocol. In addition, the address column shows the correct address, even before you send messages.
- Previously, when you used an anonymous producer to send messages to multiple addresses using the Core or AMQP protocols a producer was displayed for each address. In addition, the address shown was for the first address to which the producer sent the message, which made it appear like a normal producer sending to a single queue. In AMQ Broker 7.11, when you use an anonymous producer to send messages to multiple addresses, a single producer is displayed for each anonymous producer. In addition, the address is not connected to a specific address and has a value of **ANONYMOUS**.

Previously, the **listProducersInfoAsJSON** method provided a count of messages sent to each queue by a particular session. However, this method incorrectly returned a producer for each queue that a message was sent to. For example, if an anonymous producer sent a message to 1000 queues, this method returned 1000 producers. In AMQ Broker 7.11, the **listProducersInfoAsJSON** method now returns accurately the same data as the **listProducers** method, but in a different format.

In AMQ Broker 7.11, the following new metric data is returned:

Consumers

messagesInTransit - The number of messages for delivery which have not yet been acknowledged

messagesInTransitSize - The total size of messages for delivery which have not yet been acknowledged

messagesDelivered - The number of messages delivered

messagesDeliveredSize - The total size of messages delivered

messagesAcknowledged - The total number of messages acknowledged

messagesAcknowledgedAwaitingCommit - The total number of messages in a transaction that are acknowledged but awaiting a commit

lastDeliveredTime - The time in milliseconds of the last message delivered

lastAcknowledgedTime - The time in milliseconds of the last message acknowledged

Producers

msgSent - The number of messages sent by a producer

msgSizeSent - The total size of messages sent by a producer

lastProducedMessageID - The ID of the last message sent

CHAPTER 4. DEPRECATED FEATURES

This section describes features that are supported, but have been deprecated from AMQ Broker.

upgrade attribute in Custom Resource

Starting in 7.11, the **upgrade** attribute and the associated **enabled** and **minor** attributes are deprecated because they cannot work as originally designed. Use the **image** or **version** attributes to deploy specific broker container images.

queues configuration element

Starting in 7.10, the <queues> configuration element is deprecated. You can use the <addresses> configuration element to create addresses and associated queues. The <queues> configuration element will be removed in a future release.

getAddressesSettings method

Starting in 7.10, the get**Addresses**Settings method, which is included in the org.apache.activemq.artemis.core.config.Configuration interface, is deprecated. Use the get**Address**Settings method to configure addresses and queues for the broker programatically.

OpenWire protocol

Starting in 7.9, the OpenWire protocol is a deprecated feature. If you are creating a new AMQ Broker-based system, use one of the other supported protocols. This feature will be removed in 7.12.

Adding users when broker instance is not running

Starting in 7.8, when an AMQ Broker instance is not running, the ability to add users to the broker from the CLI interface is removed.

Network pinger

Starting in 7.5, network pinging is a deprecated feature. Network pinging cannot protect a broker cluster from network isolation issues that can lead to irrecoverable message loss. This feature will be removed in a future release. Red Hat continues to support existing AMQ Broker deployments that use network pinging. However, Red Hat no longer recommends use of network pinging in new deployments. For guidance on configuring a broker cluster for high availability and to avoid network isolation issues, see Implementing high availability in *Configuring AMQ Broker*.

Hawtio dispatch console plugin

Starting in 7.3, AMQ Broker no longer ships with the Hawtio dispatch console plugin, **dispatch-hawtio-console.war**. Previously, the dispatch console was used to manage AMQ Interconnect. However, AMQ Interconnect now uses its own, standalone web console.

CHAPTER 5. REMOVED FEATURES

This section describes features and functionality that have been removed from AMQ Broker.

Access to the root of the AMQ Broker web server

In previous versions of AMQ Broker, opening the root URL of the AMQ Broker web server, for example, http://localhost:8161/, in a browser window displays a landing page. The landing page has links to AMQ Management Console and AMQ Broker documentation. In 7.11, all static HTML content is removed from AMQ Broker. Therefore, if you open the root URL of the AMQ Broker web server, a landing page is not displayed. Instead, your browser session is automatically redirected to AMQ Management Console.

CHAPTER 6. FIXED ISSUES

For a complete list of issues that have been fixed in the release, see AMQ Broker 7.11.0 Fixed Issues and see AMQ Broker - 7.11.x Resolved Issues for a list of issues that have been fixed in patch releases.

CHAPTER 7. FIXED COMMON VULNERABILITIES AND EXPOSURES

This section details Common Vulnerabilities and Exposures (CVEs) fixed in the AMQ Broker 7.11 release.

- **ENTMQBR-6630 - CVE-2022-1278 WildFly: possible information disclosure**
- **ENTMQBR-7397 - CVE-2022-22970 springframework: DoS via data binding to multipartFile or servlet part**
- **ENTMQBR-7398 - CVE-2022-22971 springframework: DoS with STOMP over WebSocket**
- **ENTMQBR-7005 - CVE-2022-2047 jetty-http: improver hostname input handling**
- **ENTMQBR-7640 - CVE-2022-3782 keycloak: path traversal via double URL encoding**

CHAPTER 8. KNOWN ISSUES

This section describes known issues in AMQ Broker 7.11.

- **ENTMQBR-8106 - AMQ Broker Drainer pod doesn't function properly after changing MessageMigration in CR**
  You cannot change the value of the **messageMigration** attribute in a running broker deployment. To work around this issue, you must set the required value for the **messageMigration** attribute in a new **ActiveMQ Artemis** CR and create a new broker deployment.

- **ENTMQBR-8166 - Self-signed certificate with UseClientAuth=true prevents communication of Operator with Jolokia**
  If the **useClientAuth** attribute is set to **true** in the **console** section of the **ActiveMQ Artemis** CR, the Operator is unable to configure certain features, for example, create addresses, on the broker. In the Operator log, you see an error message that ends with **remote error: tls: bad certificate**.

- **ENTMQBR-7385 - Message flops around the federation queue on slow consumers**
  If the local application consumers are very slow or unable to consume message, a message can be sent back-and-forth over a federated connection a large number of times before it is finally consumed by an application consumer.

- **ENTMQBR-7820 - [Operator] Supported versions listed in 7.11.0 OPR1 operator log are incorrect**
  The Operator logs lists support for the following AMQ Broker image versions : 7.10.0 7.10.1 7.10.2 7.11.0 7.8.1 7.8.2 7.8.3 7.9.0 7.9.1 7.9.2 7.9.3 7.9.4. The Operator actually supports AMQ Broker image versions beginning with 7.10.0.

- **ENTMQBR-7359 - Change to current handling of credential secret with 7.10.0 Operator**
  The Operator stores the administrator username and password for connecting to the broker in a secret. The default secret name is in the form ***<custom-resource-name>*-credentials-secret**. You can create a secret manually or allow the Operator to create a secret.

  If the **adminUser** and **adminPassword** attributes are configured in a Custom Resource prior to 7.10.0, the Operator updates a manually-created secret with the values of these attributes. Starting in 7.10.0, the Operator no longer updates a secret that was created manually. Therefore, if you change the values of the **adminUser** and **adminPassword** attributes in the CR, you must either:

- Update the secret with the new username and password

- Delete the secret and allow the Operator to create a secret. When the Operator creates a secret, it adds the values of the **adminUser** and **adminPassword** attributes if these are specified in the CR. If these attributes are not in the CR, the Operator generates random credentials for the secret.

- **ENTMQBR-7111 - 7.10 versions of operator tend to remove StatefulSet during upgrade**
  If you are upgrading to or from AMQ Broker Operator 7.10.0, the new Operator automatically deletes the existing StatefulSet for each deployment during the reconciliation process. When the Operator deletes the StatefulSet, the existing broker pods are deleted, which causes a temporary broker outage.

  You can work around this issue by running the following command to manually delete the StatefulSet and orphan the running pods before the Operator gets to delete the StatefulSet: oc delete statefulset *<statefulset-name>* --cascade=orphan

Manually deleting the StatefulSet during the upgrade process allows the new Operator to reconcile the StatefulSet without deleting the running pods. For more information, see Upgrading the Operator using OperatorHub in *Deploying AMQ Broker on OpenShift*.

- **ENTMQBR-6473 - Incompatible configuration due to schema URL change**
  When you try to use a broker instance configuration from a previous release with a version 7.9 or 7.10 instance, an incompatible configuration as a result of a schema URL change causes the broker to crash. To work around this issue, update the schema URL in the relevant configuration files as outlined in Upgrading from 7.9.0 to 7.10.0 on Linux .

- **ENTMQBR-4813 AsynchronousCloseException with large messages and multiple C++ subscribers**
  If multiple C++ Publisher clients that uses the AMQP protocol are running on the same host as subscribers and the broker, and a publisher sends a large message, one of the subscribers crashes.

- **ENTMQBR-5749 - Remove unsupported operators that are visible in OperatorHub**
  Only the Operators and Operator channels mentioned in Deploying the Operator from OperatorHub are supported. For technical reasons associated with Operator publication, other Operator and channels are visible in the OperatorHub and should be ignored. For reference, the following list shows which Operators are visible, but not supported:
  - Red Hat Integration - AMQ Broker LTS - all channels
  - Red Hat Integration - AMQ Broker - alpha, current, and current-76

- **ENTMQBR-569 - Conversion of IDs from OpenWire to AMQP results in sending IDs as binary**
  When communicating cross-protocol from an A-MQ 6 OpenWire client to an AMQP client, additional information is encoded in the application message properties. This is benign information used internally by the broker and can be ignored.

- **ENTMQBR-655 - [AMQP] Unable to send message when populate-validated-user is enabled**
  The configuration option **populate-validated-user** is not supported for messages produced using the AMQP protocol.

- **ENTMQBR-1875 - [AMQ 7, ha, replicated store] backup broker appear not to go "live" or shutdown after - ActiveMQIllegalStateException errorType=ILLEGAL_STATE message=AMQ119026: Backup Server was not yet in sync with live**
  Removing the paging disk of a primary broker while a backup broker is trying to sync with the primary broker causes the primary to fail. In addition, the backup broker cannot become live because it continues trying to sync with the primary broker.

- **ENTMQBR-2068 - some messages received but not delivered during HA fail-over, fail-back scenario**
  Currently, if a broker fails over to its backup while an OpenWire client is sending messages, messages being delivered to the broker when failover occurs could be lost. To work around this issue, ensure that the broker persists the messages before acknowledging them.

- **ENTMQBR-3331 - Stateful set controller can’t recover from CreateContainerError, blocking the operator**
  If the AMQ Broker Operator creates a stateful set from a Custom Resource (CR) that has a configuration error, the stateful set controller is unable to roll out the updated stateful set when the error is resolved.

For example, a misspelling in the value of the **image** attribute in your main broker CR causes the status of the first Pod created by the stateful set controller to remain **Pending**. If you then fix the misspelling and apply the CR changes, the AMQ Broker Operator updates the stateful set. However, a Kubernetes known issue prevents the stateful set controller from rolling out the updated stateful set. The controller waits indefinitely for the Pod that has a **Pending** status to become **Ready**, so the new Pods are not deployed.

To work around this issue, you must delete the Pod that has a **Pending** status to allow the stateful set controller to deploy the new Pods. To check which Pod has a **Pending** status, use the following command: **oc get pods --field-selector=status.phase=Pending**. To delete a Pod, use the **oc delete pod <pod name>** command.

- ENTMQBR-3846 - **MQTT client does not reconnect on broker restart**
  When you restart a broker, or a broker fails over, the active broker does not restore connections for previously-connected MQTT clients. To work around this issue, to reconnect an MQTT client, you need to manually call the **subscribe()** method on the client.

- ENTMQBR-4127 - **AMQ Broker Operator: Route name generated by Operator might be too long for OpenShift**
  For each broker Pod in an Operator-based deployment, the default name of the Route that the Operator creates for access to the AMQ Broker management console includes the name of the Custom Resource (CR) instance, the name of the OpenShift project, and the name of the OpenShift cluster. For example, **my-broker-deployment-wconsj-0-svc-rte-my-openshift-project.my-openshift-domain**. If some of these names are long, the default Route name might exceed the limit of 63 characters that OpenShift enforces. In this case, in the OpenShift Container Platform web console, the Route shows a status of **Rejected**.

  To work around this issue, use the OpenShift Container Platform web console to manually edit the name of the Route. In the console, click the Route. On the **Actions** drop-down menu in the top-right corner, select **Edit Route**. In the YAML editor, find the **spec.host** property and edit the value.

- ENTMQBR-4140 - **AMQ Broker Operator: Installation becomes unusable if storage.size is improperly specified**
  If you configure the **storage.size** property of a Custom Resource (CR) instance to specify the size of the Persistent Volume Claim (PVC) required by brokers in a deployment for persistent storage, the Operator installation becomes unusable if you do not specify this value properly. For example, suppose that you set the value of **storage.size** to **1** (that is, without specifying a unit). In this case, the Operator cannot use the CR to create a broker deployment. In addition, even if you remove the CR and deploy a new version with **storage.size** specified correctly, the Operator still cannot use this CR to create a deployment as expected.

  To work around this issue, first stop the Operator. In the OpenShift Container Platform web console, click **Deployments**. For the Pod that corresponds to the AMQ Broker Operator, click the **More options** menu (three vertical dots). Click **Edit Pod Count** and set the value to **0**. When the Operator Pod has stopped, create a new version of the CR with **storage.size** correctly specified. Then, to restart the Operator, click **Edit Pod Count** again and set the value back to **1**.

- ENTMQBR-4141 - **AMQ Broker Operator: Increasing Persistent Volume size requires manual involvement even after recreating Stateful Set**
  If you try to increase the size of the Persistent Volume Claim (PVC) required by brokers in a deployment for persistent storage, the change does not take effect without further manual steps. For example, suppose that you configure the **storage.size** property of a Custom Resource (CR) instance to specify an initial size for the PVC. If you modify the CR to specify a *different* value of **storage.size**, the existing brokers continue to use the original PVC size. This is

the case even if you scale the deployment down to zero brokers and then back up to the original number. However, if you scale the size of the deployment up to add additional brokers, the new brokers use the new PVC size.

To work around this issue, and ensure that all brokers in the deployment use the same PVC size, use the OpenShift Container Platform web console to expand the PVC size used by the deployment. In the console, click **Storage → Persistent Volume Claims**. Click your deployment. On the **Actions** drop-down menu in the top-right corner, select **Expand PVC** and enter a new value.

CHAPTER 9. IMPORTANT LINKS

- Red Hat AMQ Broker 7.10 Release Notes
- Red Hat AMQ Broker 7.9 Release Notes
- Red Hat AMQ Broker 7.8 Release Notes
- Red Hat AMQ Broker 7.7 Release Notes
- Red Hat AMQ Broker 7.6 Release Notes
- Red Hat AMQ Broker 7.1 to 7.5 Release Notes (aggregated)
- Red Hat AMQ 7 Supported Configurations
- Red Hat AMQ 7 Component Details

Revised on 2024-04-18 15:50:19 UTC