Red Hat AMQ Broker 7.10

Release Notes for Red Hat AMQ Broker 7.10

Release Notes for AMQ Broker

Last Updated: 2024-06-10

Red Hat AMQ Broker 7.10 Release Notes for Red Hat AMQ Broker 7.10

Release Notes for AMQ Broker

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Broker 7.10 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 7.10

AMQ Broker 7.10 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 7.10 LTS version supports:

- Red Hat Enterprise Linux 7 and 8
- 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 .

NOTE

At a minimum, AMQ Broker 7.10 requires Java version 11 to run.

CHAPTER 3. NEW AND CHANGED FEATURES

This section describes a highlighted set of enhancements and new features in AMQ Broker 7.10.

High availability replication improvements

In previous versions of AMQ Broker, to use a replication high availability (HA) policy, you required at least three live-backup broker pairs. The three pairs were required so either broker in a pair could obtain a majority quorum vote and avoid a scenario where both brokers became live at the same time. Starting in 7.10, you can configure brokers to use the Apache Zookeeper coordination service to coordinate each live-backup broker pair, which eliminates the need to have at least three live-backup pairs. For more information, see Configuring a broker cluster for replication high availability using the ZooKeeper coordination service in *Configuring AMQ Broker*.

Client connection partitioning

In previous releases, there was no method to partition client connections on the server-side. Starting in 7.10, you can partition client connections, which involves routing connections for individual clients to the same broker each time the client initiates a connection. Two use cases for partitioning client connections are:

- Partitioning clients of durable subscriptions to ensure that a subscriber always connects to the broker where the durable subscriber queue is located.
- Minimizing the need to move data between brokers by attracting clients to data where it originates, also known as data gravity.
  To learn more about partitioning client connections, see Partitioning client connections in *Configuring AMQ Broker*.

AMQ Management Console authentication

To authenticate users with the AMQ Management Console, you can configure certificate-based authentication. To learn how to configure certificate-based authentication, see Configuring the broker console to use certificate-based authentication in *Configuring AMQ Broker*.

Controlling pod placement on nodes

In an operator-based broker deployment, you can configure a Custom Resource (CR) to control the placement of AMQ Broker pods on OpenShift Container Platform nodes by using node selectors, node affinity rules and taints and tolerations. For more information, see Controlling placement of broker pods on OpenShift Container Platform nodes in *Deploying AMQ Broker on OpenShift*.

Broker health checks

In an operator-based broker deployment, you can configure periodic health checks on a running broker container by using liveness and readiness probes. A liveness probe checks if the broker is running by pinging the broker's HTTP port. A readiness probe checks if the broker can accept network traffic by opening a connection to each of the acceptor ports configured for the broker. To learn how to configure health checks, see Configuring broker health checks in *Deploying AMQ Broker on OpenShift*.

Overriding the default memory limit

In an operator-based broker deployment, you can override the default memory limit that is set for a broker. By default, a broker is assigned half of the maximum memory that is available to the broker's Java Virtual Machine. To learn how to override the default memory limit, see Overriding the default memory limit for a broker in *Deploying AMQ Broker on OpenShift*.

Requesting a storage class in a Persistent Volume Claim (PVC)

By default, any Persistent Volume Claim (PVC) by AMQ Broker on OpenShift Container Platform uses the default storage class that is configured for the cluster. With this release, you can configure a CR to specify a storage class for AMQ Broker. To learn how to specify a storage class in a PVC, see

Configuring broker storage size and storage class in *Deploying AMQ Broker on OpenShift*.

Configuring a security context for a pod

In an operator-based broker deployment, you can configure a security context for a pod. A security context defines privilege and access control settings for a pod and includes attributes for discretionary access control, Security Enhanced Linux (SELinux), Secure Computing Mode (seccomp), sysctl interface, and Window's specific attributes for container running on Windows. For more information, see Custom Resource configuration reference in *Deploying AMQ Broker on OpenShift*.

Support for OpenShift Container Platform 4.15

In addition to supporting OpenShift Container Platform 4.6, 4.7, 4.8, 4.9 and 4.10, AMQ Broker now supports OpenShift Container Platform 4.15.

Change the default service account name for a broker pod

You can change the default service account name for a broker pod by using the **serviceAccountName** name attribute. For more information, see Custom Resource configuration reference in *Deploying AMQ Broker on OpenShift*.

Labeling broker pods

You can assign labels to broker pods by using the **labels** attribute. For more information, see Custom Resource configuration reference in *Deploying AMQ Broker on OpenShift*.

Update the acceptor and connector configuration with *StoreType and *StoreProvider

In the CR configuration for acceptors and connectors, you can specify details of the keystore and truststore that the broker uses.

Operator channels

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

- **7.10.x** - This channel provides updates for version 7.10 only and is the current Long Term Support (LTS) channel.
- **7.x** - Currently, this channel provides updates for version 7.9 only.
- **7.8.x** - This channel provides updates for version 7.8 only and was the previous 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.

Use a wildcard value to grant access to all domains using the Management API

Starting in 7.10.1, in the **management.xml** file, you can specify a wildcard value in the **entry domain** field. When you access the management API, a wildcard value in the **entry domain** field grants access to all domains.

```
<authorisation>
  <allowlist>
    <entry domain="*"/>
```

```
</allowlist>
```

JGroups 5.x

Previous versions of AMQ Broker used JGroups 3.x. AMQ Broker 7.10 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 7.10.

CHAPTER 4. DEPRECATED FEATURES

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

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 a future release.

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

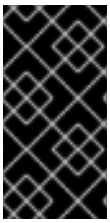
Starting in 7.5, network pinging is a deprecated feature. Network pinging cannot protect a broker cluster from network isolation issues that can lead to irrecoverable message loss. This feature will be removed in a future release. Red Hat continues to support existing AMQ Broker deployments that use 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. TECHNOLOGY PREVIEW

This section describes Technology Preview features in AMQ Broker 7.10.



IMPORTANT

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 for production. For more information, see Red Hat Technology Preview Features Support Scope.

New attributes for configuring address limits for paging

You can configure the following new attributes to set individual and global address limits based on the number of messages.

max-size-messages is the maximum number of messages allowed for the address before the broker executes the policy specified for the address-full-policy. The default value is -1, which means that there is no message limit.

global-max-messages is the total number of messages that the broker can use for all addresses. When this limit is reached, for the address associated with an incoming message, the broker executes the policy that is specified as a value for the address-full-policy. The default value is -1, which means that there is no message limit.

NOTE

If limits that are set for the **max-size-bytes** or **global-max-size** attributes are reached before the limits set for the **max-size-message** or **global-max-messages** attributes, the broker executes the address-full policy.

CHAPTER 6. FIXED ISSUES

For a complete list of issues that have been fixed in the release, see link: AMQ Broker 7.10.0 Fixed Issues and see AMQ Broker - 7.10.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 7.10 release.

- ENTMQBR-5140 - CVE-2019-10744 nodejs-lodash: prototype pollution in defaultsDeep function leading to modifying properties
- ENTMQBR-5893 - CVE-2021-4040 broker: AMQ Broker: Malformed message can result in partial DoS (OOM)
- ENTMQBR-5933 - CVE-2021-43797 netty: control chars in header names may lead to HTTP request smuggling
- ENTMQBR-6401 - CVE-2022-23913 artemis-commons: Apache ActiveMQ Artemis DoS
- ENTMQBR-6477 - CVE-2020-36518 jackson-databind: denial of service via a large depth of nested objects

CHAPTER 8. KNOWN ISSUES

This section describes known issues in AMQ Broker 7.10.

- 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-7363 - **redeliveryDelayMultiplier in AddressSettingsType from 7.9 CR cannot be reconciled**
  If the **redeliveryDelayMultiplier** and the **redeliveryCollisionAvoidanceFactor** attributes are configured in the main broker CR in a 7.8.x or 7.9.x deployment, the new Operator is unable to reconcile any CR after you upgrade to 7.10.x. The reconcile fails because the data type of both attributes changed from float to string in 7.10.x.

  You can work around this issue by deleting the **redeliveryDelayMultiplier** and the **redeliveryCollisionAvoidanceFactor** attributes from the **spec.deploymentPlan.addressSettings.addressSetting** element. Then, configure the attributes in the **brokerProperties** element. For example:

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

NOTE

In the **brokerProperties** element, use the **redeliveryMultiplier** attribute name instead of the **redeliveryDelayMultiplier** attribute name that you deleted.

- ENTMQBR-7396 - **[Operator, upgrade] the upgrade to 7.10.1 fails to create new Acceptor/Connector *v1.ServiceAdmission**
  After you upgrade from AMQ Broker 7.10.0 to 7.10.1, messaging stops working if incorrect labels added to services and Pod selectors in 7.10.0 were not removed during the upgrade. If messaging is not working after the upgrade, complete the following steps to resolve this issue.

  1. Log in to the OpenShift Container Platform web console as a cluster administrator.
  2. From the Project drop-down menu at the top of the page, select the project in which the Operator is installed.

3. In the left navigation menu, click **Networking → Services**.
4. In the **Labels** and **Pod Selectors** columns, check if any of the services has labels other than **ActiveMQArtemis** and **application** configured.
5. For each service that has labels other than **ActiveMQArtemis** and **application** configured, complete the following steps to remove the labels.
   a. Click the service to open the **Details** tab.
   b. In the **Labels** field, click **Edit** and delete all labels apart from the **ActiveMQArtemis** and **application** labels.
   c. Click **Save**.
   d. Click the **YAML** tab.
   e. In the **selector** element, delete all labels except the **ActiveMQArtemis**, **application** and **statefulset.kubernetes.io/podname** labels.
   f. Click **Save**.

- 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-6991 - **7.10-opr-3 does not fix the PV ownerRef for 7.10-opr-2 users**
  If you deploy or upgrade to 7.10.0-opr-2 and scale up your deployment, the new PVs are created with an **ownerReference** attribute that can cause data loss if you later delete the deployment CR. For example, if you deploy 7.10.0-opr-1, upgrade to 7.10.0-opr-2 and then scale from 3 to 4 broker instances, you could lose data if you delete the **ActiveMQArtemis** CR.

  To workaround this issue, you can:

  - Skip the 7.10.0-opr-2 upgrade if possible.
  - Avoid scaling up your deployment while the 7.10.0-opr-2 release is active in your cluster. You can scale up after you deploy 7.10.0-opr-3.
  - Avoid deleting the deployment CR until a later release resolves this issue.
  - Manually remove the **ownerReference** value for the PVs affected.

- ENTMQBR-6712 - **Taints and Tolerations - "tolerationSeconds" breaks the deployment**
  If you add a **tolerationSeconds** attribute in the **tolerations** section of the CR, the Operator reconciliation process does not work, and broker pods are not scheduled correctly. To work around this issue, do not add a **tolerationSeconds** attribute in the **tolerations** section of the

CR.

- 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-6655 - **The command artemis check queue fails with "Could not start Jolokia agent"**
  Before it runs, the **artemis check queue** command displays the following error message: **Could not start Jolokia agent: java.lang.IllegalStateException: Cannot open keystore for https communication: java.net.BindException: Address already in use**.

- ENTMQBR-6654 - **requireLogin:true works only for a new broker CR applied and not for existing one.**
  If the **requireLogin** property is set to **true** in the CR, the AMQ_REQUIRE_LOGIN environment variable is not updated in the stateful set of existing broker instances and the console credential are not validated. To work around this issue, manually update the environment variable value in the stateful set for the existing instance.

- ENTMQBR-5936 - **The client doesn't failover to the backup server if the URL targets non clustered ports.**
  If the connection URL that a client uses to connect to a HA cluster has a port that is not configured in the broker's **static-connectors**, after a failover occurs, the client retries the connection to the previously live broker and does not attempt to connect to the new live broker.

- ENTMQBR-6728 - **Upgrade path is broken**
  This issue prevents AMQ Broker 7.9 users who are subscribed to the **7.x** channel from automatically upgrading to AMQ Broker 7.10. To workaround this issue, subscribe to the **7.10.x** channel.

- 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-17 - **AMQ222117: Unable to start cluster connection**
  A broker cluster may fail to initialize properly in environments that support IPv6. The failure is due to a **SocketException** that is indicated by the log message **Can't assign requested address**. To work around this issue, set the **java.net.preferIPv4Stack** system property to **true**.

- ENTMQBR-520 - **Receiving from address named the same as a queue bound to another address should not be allowed**

A queue with the same name as an address must only be assigned to address. Creating a queue with the same name as an existing address, but bound to an address with a different name, is an invalid configuration. Doing so can result in incorrect messages being routed to the queue.

- 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-636 - **Journal breaks, causing `JavaNullPointerException`, under perf load (mpt)**
  To prevent IO-related issues from occurring when the broker is managing heavy loads, verify that the JVM is allocated with enough memory and heap space. See the section titled "Tuning the VM" in the Performance Tuning chapter of the ActiveMQ Artemis documentation.

- ENTMQBR-648 - **JMS Openwire client is unable to send messages to queue with defined `purgeOnNoConsumer` or queue filter**
  Using an A-MQ 6 JMS client to send messages to an address that has a queue with **`purgeOnNoConsumer`** set to **`true`** fails if the queue has no consumers. It is recommended that you do not set the **`purgeOnNoConsumer`** option when using A-MQ 6 JMS clients.

- ENTMQBR-652 - **List of known `amq-jon-plugin` bugs**
  This version of **`amq-jon-plugin`** has known issues with the MBeans for broker and queue.

  Issues with the broker MBean:

  - Closing a connection throws **`java.net.SocketTimeoutException`** exception
  - **`listSessions()`** throws **`java.lang.ClassCastException`**
  - Adding address settings throws **`java.lang.IllegalArgumentException`**
  - **`getConnectorServices()`** operation cannot be found
  - **`listConsumersAsJSON()`** operation cannot be found
  - **`getDivertNames()`** operation cannot be found
  - Listing network topology throws **`IllegalArgumentException`**
  - Remove address settings has wrong parameter name

  Issues with the queue MBean:

  - **`expireMessage()`** throws argument type mismatch exception
  - **`listDeliveringMessages()`** throws **`IllegalArgumentException`**
  - **`listMessages()`** throws **`java.lang.Exception`**
  - **`moveMessages()`** throws **`IllegalArgumentException`** with error message argument type mismatch
  - **`removeMessage()`** throws **`IllegalArgumentException`** with error message argument type mismatch

- **removeMessages()** throws exception with error Can't find operation removeMessage with 2 arguments
- **retryMessage()** throws argument type mismatch **IllegalArgumentException**

- 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-897 - **Openwire client/protocol issues with special characters in destination name**
  Currently AMQ OpenWire JMS clients cannot access queues and addresses that include the following characters in their name: comma (','), hash ('#'), greater than ('>'), and whitespace.

- ENTMQBR-944 - **[A-MQ7, Hawtio, RBAC] User gets no feedback if operation access was denied by RBAC**
  The console can indicate that an operation attempted by an unauthorized user was successful when it was not.

- 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 master broker while a backup broker is trying to sync with the master broker causes the master to fail. In addition, the backup broker cannot become live because it continues trying to sync with the master.

- ENTMQBR-2068 - **some messages received but not delivered during HA fail-over, fail-back scenario**
  Currently, if a broker fails over to its slave 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-4023 - **AMQ Broker Operator: Pod Status pod names do not reflect the reality**

For an Operator-based broker deployment in a given OpenShift project, if you use the **oc get pod** command to list the broker Pods, the ordinal values for the Pods start at **0**, for example, **amq-operator-test-broker-ss-0**. However, if you use the **oc describe** command to get the status of broker Pods created from the **activemqartmises** Custom Resource (that is, **oc describe activemqartemises**), the Pod ordinal values incorrectly start at **1**, for example, **amq-operator-test-broker-ss-1**. There is no way to work around this issue.

- 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 7.9 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 2024-06-10 15:28:56 UTC