Red Hat AMQ 2021.q3

Release Notes for AMQ Streams 1.8 on OpenShift

For use with AMQ Streams on OpenShift Container Platform

Last Updated: 2021-12-14

Red Hat AMQ 2021.q3 Release Notes for AMQ Streams 1.8 on OpenShift

For use with AMQ Streams on OpenShift Container Platform

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Streams 1.8 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. FEATURES

AMQ Streams version 1.8 is based on Strimzi 0.24.x.

The features added in this release, and that were not in previous releases of AMQ Streams, are outlined below.

NOTE

To view all the enhancements and bugs that are resolved in this release, see the AMQ Streams Jira project.

1.1. OPENSHIFT CONTAINER PLATFORM SUPPORT

AMQ Streams 1.8 is supported on OpenShift Container Platform 4.6 and 4.8.

For more information about the supported platform versions, see the Red Hat Knowledgebase article Red Hat AMQ 7 Supported Configurations .

1.2. KAFKA 2.8.0 SUPPORT

AMQ Streams now supports Apache Kafka version 2.8.0.

AMQ Streams uses Kafka 2.8.0. Only Kafka distributions built by Red Hat are supported.

You must upgrade the Cluster Operator to AMQ Streams version 1.8 before you can upgrade brokers and client applications to Kafka 2.8.0. For upgrade instructions, see Upgrading AMQ Streams .

Refer to the Kafka 2.7.0 and Kafka 2.8.0 Release Notes for additional information.

NOTE

Kafka 2.7.x is supported only for the purpose of upgrading to AMQ Streams 1.8.

For more information on supported versions, see the Red Hat AMQ 7 Component Details Page on the Customer Portal.

Kafka version 2.8.0 requires ZooKeeper version 3.5.9. Therefore, the Cluster Operator performs a ZooKeeper upgrade when you upgrade from AMQ Streams 1.7 to AMQ Streams 1.8.

1.3. FEATURE GATES TO TOGGLE FEATURES ON AND OFF

As a Kafka cluster administrator, you can now toggle a subset of features on and off using *feature gates* in the operator's deployment configuration. Feature gates are currently available only for the Cluster Operator; future releases might add feature gates to other operators.

AMQ Streams 1.8 introduces the following feature gates and associated new features:

- **ControlPlaneListener** to toggle control plane listeners
- **ServiceAccountPatching** to toggle service account patching

Feature gates have a default state of *enabled* or *disabled*. When enabled, a feature gate changes the behavior of the operator and enables the feature in your AMQ Streams deployment.

Feature gates have a maturity level of *Alpha*, *Beta*, or *Generally Available* (GA).

Table 1.1. Maturity levels of feature gates

Feature gate maturity level	Description	Default state
Alpha	Controls features that might be experimental, unstable, or not sufficiently tested for production use. These features are subject to change in future releases.	Disabled
Beta	Controls features that are well tested. These features are not likely to change in future releases.	Enabled
General Availability (GA)	Controls features that are stable, well tested, and suitable for production use. GA features will not change in future releases.	Enabled

Configuring feature gates

In the Cluster Operator's deployment configuration, in the **STRIMZI_FEATURE_GATES** environment variable, specify a comma-separated list of feature gate names and prefixes. A **+** prefix enables the feature gate and a **-** prefix disables it.

Example: Enabling the Control Plane Listener feature gate

1. Edit the Deployment for the Cluster Operator:

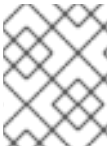
```
oc edit deployment strimzi-cluster-operator
```

2. Add the **STRIMZI_FEATURE_GATES** environment variable with a value of **+ControlPlaneListener**

```
# ...
env:
 #...
- name: STRIMZI_FEATURE_GATES
  value: +ControlPlaneListener
#...
```

See Feature gates and Cluster Operator configuration.

1.4. CONTROL PLANE LISTENERS



NOTE

This feature is controlled using the **ControlPlaneListener** feature gate, which is in alpha stage and disabled by default. For more information, see Feature gates.

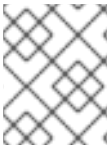
In a standard AMQ Streams cluster, control plane traffic and data plane traffic both use the same inter-broker listener on port 9091.

With this release, you can configure your cluster so that control plane traffic uses a dedicated *control plane listener* on port 9090. Data plane traffic continues to use the listener on port 9091.

Using control plane listeners might improve performance because important controller connections, such as partition leadership changes, are not delayed by data replication across brokers. The majority of data plane traffic consists of this data replication.

See Control plane listener feature gate.

1.5. SERVICE ACCOUNT PATCHING



NOTE

This feature is controlled using the **ServiceAccountPatching** feature gate, which is in alpha stage and disabled by default. For more information, see Feature gates.

By default, the Cluster Operator does not update service accounts.

With this release, you can enable updates to service accounts to be applied in every reconciliation. For example, the Cluster Operator can apply custom labels or annotations to the service account. Custom labels and annotations are configured for custom resources using the **template.serviceAccount** property.

Example custom labels and annotations

```
# ...
template:
  serviceAccount:
    metadata:
      labels:
        label1: value1
        label2: value2
      annotations:
        annotation1: value1
        annotation2: value2
# ...
```

See Service Account patching feature gate.

1.6. DEBEZIUM FOR CHANGE DATA CAPTURE INTEGRATION

Red Hat Debezium is a distributed change data capture platform. It captures row-level changes in databases, creates change event records, and streams the records to Kafka topics. Debezium is built on Apache Kafka. You can deploy and integrate Debezium with AMQ Streams. Following a deployment of AMQ Streams, you deploy Debezium as a connector configuration through Kafka Connect. Debezium passes change event records to AMQ Streams on OpenShift. Applications can read these *change event streams* and access the change events in the order in which they occurred.

Debezium has multiple uses, including:

- Data replication

- Updating caches and search indexes
- Simplifying monolithic applications
- Data integration
- Enabling streaming queries

Debezium provides connectors (based on Kafka Connect) for the following common databases:

- Db2
- MongoDB
- MySQL
- PostgreSQL
- SQL Server

For more information on deploying Debezium with AMQ Streams, refer to the product documentation.

1.7. SERVICE REGISTRY

You can use Service Registry as a centralized store of service schemas for data streaming. For Kafka, you can use Service Registry to store *Apache Avro* or JSON schema.

Service Registry provides a REST API and a Java REST client to register and query the schemas from client applications through server-side endpoints.

Using Service Registry decouples the process of managing schemas from the configuration of client applications. You enable an application to use a schema from the registry by specifying its URL in the client code.

For example, the schemas to serialize and deserialize messages can be stored in the registry, which are then referenced from the applications that use them to ensure that the messages that they send and receive are compatible with those schemas.

Kafka client applications can push or pull their schemas from Service Registry at runtime.

For more information on using Service Registry with AMQ Streams, refer to the Service Registry documentation.

CHAPTER 2. UPGRADE REQUIREMENTS

You must upgrade your custom resources to use API version **v1beta2** before upgrading to AMQ Streams version 1.8.

The **v1beta2** API version for all custom resources was introduced with AMQ Streams 1.7. For AMQ Streams 1.8, **v1alpha1** and **v1beta1** API versions were removed from all AMQ Streams custom resources apart from **KafkaTopic** and **KafkaUser**.

Upgrade of the custom resources to **v1beta2** prepares AMQ Streams for a move to Kubernetes CRD **v1**, which is required for Kubernetes v1.22.

If you are upgrading from an AMQ Streams version prior to version 1.7:

1. Upgrade to AMQ Streams 1.7
2. Convert the custom resources to **v1beta2**
3. Upgrade to AMQ Streams 1.8

See Deploying and upgrading AMQ Streams .

2.1. UPGRADING CUSTOM RESOURCES TO THE V1BETA2 VERSION

To support the upgrade of custom resources to **v1beta2**, AMQ Streams provides an *API conversion tool*, which you can download from the AMQ Streams download site .

You perform the custom resources upgrades in two steps.

Step one: Convert the format of custom resources

Using the API conversion tool, you can convert the format of your custom resources into a format applicable to **v1beta2** in one of two ways:

- Converting the YAML files that describe the configuration for AMQ Streams custom resources
- Converting AMQ Streams custom resources directly in the cluster

Alternatively, you can manually convert each custom resource into a format applicable to **v1beta2**. Instructions for manually converting custom resources are included in the documentation.

Step two: Upgrade CRDs to v1beta2

Next, using the API conversion tool with the **crd-upgrade** command, you must set **v1beta2** as the *storage* API version in your CRDs. You cannot perform this step manually.

For full instructions, see Upgrading AMQ Streams .

CHAPTER 3. ENHANCEMENTS

The enhancements added in this release are outlined below.

3.1. KAFKA 2.8.0 ENHANCEMENTS

For an overview of the enhancements introduced with Kafka 2.8.0, refer to the Kafka 2.8.0 Release Notes.

3.2. KAFKA CONNECT BUILD CONFIGURATION UPDATES

You can use **build** configuration so that AMQ Streams automatically builds a container image with the connector plugins you require for your data connections.

A dedicated service account is now created with Kafka Connect build pods. The service account is distinct from Kafka Connect itself. Before this release, the build ran under the default service account. Having its own identity is useful when specifying authentication and access.

Kafka Connect build now also works behind proxies if standard HTTP proxies (**HTTP_PROXY**, **HTTPS_PROXY**, and **NO_PROXY**) are set as environment variables for the AMQ Streams deployment.

See:

- Creating a new container image automatically using AMQ Streams
- **Build** schema reference

3.3. KUBERNETES CONFIGURATION PROVIDER FOR EXTERNAL CONFIGURATION DATA

Use the *Kubernetes Configuration Provider* plugin to load configuration data from external sources. You can load data from OpenShift Secrets or ConfigMaps.

The provider operates independently of AMQ Streams. It loads the data without needing to restart the Kafka component, even when using a new Secret or ConfigMap.

You can use it to load configuration data for all Kafka components, including producers and consumers. Use it, for example, to provide the credentials for a Kafka Connect instance hosting multiple connectors without disruption

See Loading configuration values from external sources .

3.4. LOG FILTERS WITH MARKERS

If you are using a ConfigMap to configure the (log4j2) logging levels for AMQ Streams Operators, you can now define logging filters to limit what is returned in the log. You add the filter properties to the ConfigMap.

The filters use *markers* to specify what to include in the log. You specify a kind, namespace and name for the marker. For example, if a Kafka cluster is failing, you can isolate the logs by specifying the kind as **Kafka**, and use the namespace and name of the failing cluster.

This example shows a marker filter for a Kafka cluster named **my-kafka-cluster**.

Basic logging filter configuration

```
appender.console.filter.filter1.type=MarkerFilter (1)
appender.console.filter.filter1.onMatch=ACCEPT (2)
appender.console.filter.filter1.onMismatch=DENY (3)
appender.console.filter.filter1.marker=Kafka(my-namespace/my-kafka-cluster) (4)
```

1. The **MarkerFilter** type compares a specified marker for filtering.
2. The **onMatch** property accepts the log if the marker matches.
3. The **onMismatch** property rejects the log if the marker does not match.
4. The marker used for filtering is in the format *KIND(NAMESPACE/NAME-OF-RESOURCE)*.

See Adding logging filters to Operators

3.5. OAUTH 2.0 AUTHENTICATION ENHANCEMENTS

Configure audience and scope

You can now configure the **clientAudience** and **clientScope** properties when obtaining a token from the authorization server. The property values are passed to the token endpoint as **audience** and **scope** parameters. Both properties are configured in the OAuth 2.0 authentication listener configuration in the **Kafka** custom resource.

Use these properties in the following scenarios:

- When obtaining an access token for inter-broker authentication
- In the name of a client for OAuth 2.0 over PLAIN client authentication, using a **clientId** and **secret**
  Specifically, the **audience** and **scope** can now be included in the request when the PLAIN callback first exchanges the **clientID** (as the username) and the **secret** (as the password) with the authorization server in order to obtain an access token.

These properties affect whether a client can obtain a token and the content of the token. They do not affect token validation rules imposed by the listener.

Example configuration for clientAudience and clientScope properties

```
# ...
  authentication:
    type: oauth
    # ...
    clientAudience: AUDIENCE
    clientScope: SCOPE
```

Authorization servers sometimes provide **aud** (audience) claims in JWT access tokens. When audience checks are enabled, the Kafka broker rejects tokens that do not contain the broker's **clientId** in their **aud** claims. To enable audience checks, set the **checkAudience** option to **true** in the **oauth** listener configuration. Audience checks are disabled by default.

See Configuring OAuth 2.0 support for Kafka brokers and KafkaListenerAuthenticationOAuth schema reference

Specify audience for Kafka Connect and Kafka Bridge

You can now specify the **audience** option when configuring OAuth client authentication for Kafka Connect or the Kafka Bridge in their respective custom resources. Previously, only the **scope** option was supported for these resources.

See KafkaClientAuthenticationOAuth schema reference

Token endpoint not required with OAuth 2.0 over PLAIN

The **tokenEndpointUri** option is no longer required when using the "client ID and secret" method for OAuth 2.0 over PLAIN authentication.

Example OAuth 2.0 over PLAIN configuration with token endpoint URI specified

```
# ...
  authentication:
    type: oauth
    # ...
    enablePlain: true
    tokenEndpointUri: https://OAUTH-SERVER-ADDRESS/auth/realms/external/protocol/openid-connect/token
```

If the **tokenEndpointUri** is not specified, the listener treats the:

- **username** parameter as the account name
- **password** parameter as the raw access token, which is passed to the authorization server for validation (the same behavior as for OAUTHBEARER authentication)

The behavior of the "long-lived access token" method for OAuth 2.0 over PLAIN authentication is unchanged. The **tokenEndpointUri** is not required when using this method.

See OAuth 2.0 authentication mechanisms

3.6. USER QUOTAS

The handling of user quotas through the User Operator is no longer managed by ZooKeeper. Instead, user quotas are handled through the API.

Additionally, support has been added for Kafka's mutation rate quota. This quota limits the number of partition mutations allowed per second. The quota prevents Kafka clusters from being overwhelmed by concurrent topic operations.

The number of partition mutations includes the following types of user requests:

- Creating partitions for a new topic
- Adding partitions to an existing topic
- Deleting partitions from a topic

You can configure a mutation rate quota to control the rate at which mutations are accepted for user requests. The rate is accumulated from the number of partitions created or deleted.

Use the **controllerMutationRate** option to apply the quota to the Kafka user. In this example, 10 partition creation and deletion operations are allowed per second.

Example KafkaUser configuration with user quotas

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaUser
metadata:
  name: my-user
  labels:
    strimzi.io/cluster: my-cluster
spec:
  # ...
  quotas:
    #...
    controllerMutationRate: 10 (1)
```

See User quotas

3.7. PAUSE RECONCILIATION OF CUSTOM RESOURCES

You can pause the reconciliation of custom resources managed by AMQ Streams operators to perform fixes or make updates. You can also pause reconciliation of custom resources you are creating. The custom resource is created, but it is ignored.

You add an annotation to the custom resource to pause it.

Example annotation for pausing reconciliation

```
oc annotate KIND-OF-CUSTOM-RESOURCE NAME-OF-CUSTOM-RESOURCE strimzi.io/pause-reconciliation="true"
```

It is now possible to pause reconciliation of **KafkaTopic** custom resources.

See Pausing reconciliation of custom resources

3.8. KAFKA EXPORTER UPDATE

The custom version of Kafka Exporter that is provided with AMQ Streams has been updated to version 1.3.1. AMQ Streams includes an example Grafana dashboard for Kafka Exporter in the examples provided (**examples/metrics/grafana-dashboards/strimzi-kafka-exporter.json**).

See Add Kafka Exporter

3.9. KAFKA CONNECT BUILD USES HASHES TO NAME DOWNLOAD FILES

You can configure a **KafkaConnect** resource to create a custom Kafka Connect container image. Using **spec.build** configuration automates the process. You configure **plugins** to specify the implementation artifacts and **output** to reference the container registry that stores the image. AMQ Streams downloads and adds the connector plugins into the new container image.

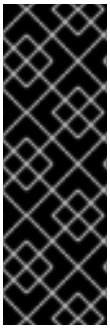
The build process now uses the URL hash to name downloaded artifact files. Previously, it used the last segment of the download URL. If your plugin artifact requires a specific name, you can use a new **other** artifact type and its **fileName** field.

Example naming of a plugin artifact

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      #...
    plugins:
      - name: my-plugin
        artifacts:
          - type: other
            url: https://my-domain.tld/my-other-file.ext
            sha512sum: 589...ab4
            fileName: name-of-file.ext
  #...
```

See **Build** schema reference

CHAPTER 4. TECHNOLOGY PREVIEWS



IMPORTANT

Technology Preview features are not supported with Red Hat production service-level agreements (SLAs) and might not be functionally complete; therefore, Red Hat does not recommend implementing any Technology Preview features in production environments. This Technology Preview feature provides early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. For more information about support scope, see Technology Preview Features Support Scope.

4.1. KAFKA STATIC QUOTA PLUGIN CONFIGURATION

Use the *Kafka Static Quota* plugin to set throughput and storage limits on brokers in your Kafka cluster. You enable the plugin and set limits by configuring the **Kafka** resource. You can set a byte-rate threshold and storage quotas to put limits on the clients interacting with your brokers.

Example Kafka Static Quota plugin configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    config:
      client.quota.callback.class: io.strimzi.kafka.quotas.StaticQuotaCallback
      client.quota.callback.static.produce: 1000000
      client.quota.callback.static.fetch: 1000000
      client.quota.callback.static.storage.soft: 400000000000
      client.quota.callback.static.storage.hard: 500000000000
      client.quota.callback.static.storage.check-interval: 5
```

See Setting limits on brokers using the Kafka Static Quota plugin

4.2. CRUISE CONTROL FOR CLUSTER REBALANCING

NOTE

Cruise Control remains in Technology Preview, with some new enhancements.

You can deploy Cruise Control and use it to rebalance your Kafka cluster using *optimization goals* — defined constraints on CPU, disk, network load, and more. In a balanced Kafka cluster, the workload is more evenly distributed across the broker pods.

Cruise Control is configured and deployed as part of a **Kafka** resource. You can use the default optimization goals or modify them to suit your requirements. Example YAML configuration files for Cruise Control are provided in **examples/cruise-control/**.

When Cruise Control is deployed, you can create **KafkaRebalance** custom resources to:

- Generate optimization proposals from multiple optimization goals
- Rebalance a Kafka cluster based on an optimization proposal

Other Cruise Control features are not currently supported, including anomaly detection, notifications, write-your-own goals, and changing the topic replication factor.

See Cruise Control for cluster rebalancing

4.2.1. Enhancements to the Technology Preview

Refresh optimization proposals

You can now reuse existing **KafkaRebalance** resources with a status of **Ready**, which indicates that cluster rebalancing completed successfully. You can reuse the optimization goals defined in the **KafkaRebalance** resource, or change the goals.

To refresh an optimization proposal:

1. Check the status of the **KafkaRebalance** resource:

```
oc describe kafkarebalance REBALANCE-NAME
```

2. Apply the **strimzi.io/rebalance=refresh** annotation:

```
oc annotate kafkarebalance REBALANCE-NAME strimzi.io/rebalance=refresh
```

Cruise Control refreshes the optimization proposal to reflect the latest state of your Kafka cluster.

See Approving an optimization proposal

View the broker load in optimization proposals

Optimization proposals now consist of the *broker load* in addition to the summary status. The broker load is returned inside a ConfigMap and shows metrics for the load on each Kafka broker, including the CPU utilization, disk usage, network output rate, and more. The metrics are presented in three categories:

before
: Current value before the optimization proposal is applied

after
: Expected value after the optimization proposal is applied

difference
: Difference between the after value and the before value

The broker load ConfigMap has the same name as the **KafkaRebalance** resource. The metrics are encoded as a JSON string. To view them in a human readable format, use **jq** or a similiar JSON parser. For example:

```
oc get configmap MY-REBALANCE -o json | jq '.["data"]["brokerLoad.json"]|fromjson|.'
```

See Optimization proposals overview

CHAPTER 5. DEPRECATED FEATURES

The features deprecated in this release, and that were supported in previous releases of AMQ Streams, are outlined below.

5.1. KAFKA CONNECT WITH SOURCE-TO-IMAGE (S2I)

With the introduction of **build** configuration to the KafkaConnect resource, AMQ Streams can now automatically build a container image with the connector plugins you require for your data connections.

As a result, support for Kafka Connect with Source-to-Image (S2I) is deprecated.

To prepare for this change, you can migrate Kafka Connect S2I instances to Kafka Connect instances.

See Migrating from Kafka Connect with S2I to Kafka Connect

5.2. ENABLEECDSA PROPERTY

The **enableECDSA** property has been deprecated in the **oauth** listener configuration for the Kafka broker. The value of this property is now ignored.

ECDSA encryption is available through the Java Cryptography Extension (JCE). The Bouncy Castle Crypto libraries are no longer packaged with AMQ Streams.

5.3. INCLUSIVE LANGUAGE

The **topicsBlacklistPattern** and **groupsBlacklistPattern** properties in the **KafkaMirrorMaker2** resource are deprecated. The properties will be removed and replaced by **topicsExcludePattern** and **groupsExcludePattern**.

The **whitelist** property in the **KafkaMirrorMaker** resource is deprecated. The property will be replaced by **include**.

5.4. KAFKA CONNECT NAMING OF PLUGIN ARTIFACT FILES

The Kafka Connect build process now uses the URL hash to name downloaded artifact files. Previously, it used the last segment of the download URL.

See Kafka Connect build uses hashes to name download files

5.5. DEPRECATED AND REMOVED KAFKA FEATURES

This section gives advance notice of important deprecations and removals in the Apache Kafka project.

5.5.1. Planned for removal in Kafka version 3.0

Kafka version 3.0 will be shipped with the next major release of AMQ Streams.

The following table shows methods and components that were deprecated in Kafka 2.x or earlier and will be **removed** in Kafka 3.0. This list is not exhaustive.

Table 5.1. Deprecated API methods and components that will be removed in Kafka 3.0

API or component	Issue link	Description
Admin API	KAFKA-12581	Remove deprecated Admin.electPreferredLeaders
Admin API	KAFKA-6987	Reimplement KafkaFuture with CompletableFuture (deprecate KafkaFuture.Function)
Admin client	KAFKA-12577	Remove deprecated **ConfigEntry** constructor
All clients	KAFKA-12579	Remove various deprecated methods from clients for 3.0
All clients	KAFKA-12600	Remove deprecated config value **default** for client config **client.dns.lookup**
All clients	KAFKA-12578	Remove deprecated security classes/methods
Broker	KAFKA-12591	Remove deprecated **quota.producer.default** and **quota.consumer.default** configurations
Broker	KAFKA-12592	Remove deprecated LogConfig.Compact
Broker	KAFKA-12590	Remove deprecated SimpleAclAuthorizer
Broker	KAFKA-5905	Remove PrincipalBuilder and DefaultPrincipalBuilder
Common	KAFKA-12573	Removed deprecated **Metric#value**
Consumer API	KAFKA-12637	Remove deprecated PartitionAssignor interface
Connect API	KAFKA-12482	Remove deprecated rest.host.name and rest.port Connect worker configs
Connect API	KAFKA-12945	Remove port, host.name, and related configs in 3.0

API or component	Issue link	Description
Connect API	KAFKA-12717	Remove internal converter config properties
Streams API	KAFKA-12574	Deprecate eos-alpha
Streams API	KAFKA-12808	Remove deprecated methods under StreamsMetrics
Streams API	KAFKA-7606	Remove deprecated options from StreamsResetter
Streams API	KAFKA-12796	Removal of deprecated classes under **streams-scala**
Streams API	KAFKA-12419	Remove deprecated APIs of Kafka Streams in 3.0
Streams API	KAFKA-10434	Remove deprecated methods on WindowStore
Streams API	KAFKA-12449	Remove deprecated WindowStore#put
Streams API	KAFKA-12813	Remove deprecated schedule method in ProcessorContext
Streams API	KAFKA-12809	Remove deprecated methods under Stores
Streams API	KAFKA-12814	Remove deprecated method StreamsConfig#getConsumerConfig
Streams API	KAFKA-12313	Deprecate the default.windowed.serde.inner.class configs
Streams API	KAFKA-8372	Remove deprecated RocksDB#compactRange API
Streams API	KAFKA-12584	Remove deprecated **Sum** and **Total** classes
Streams API	KAFKA-12683	Remove deprecated "UsePreviousTimeOnInvalidTimeStamp"

API or component	Issue link	Description
Streams API	KAFKA-12810	Remove deprecated TopologyDescription.Source#topics
Streams API	KAFKA-12630	Remove deprecated KafkaClientSupplier#getAdminClient
Streams API	KAFKA-10046	Deprecated PartitionGrouper config is ignored
Streams API	KAFKA-12633	Remove deprecated "TopologyTestDriver#pipeInput / readOutput"
Streams API	KAFKA-12441	Remove deprecated methods StreamsBuilder#addGlobalStore
Streams API	KAFKA-12452	Remove deprecated overloads for ProcessorContext#forward
Streams API	KAFKA-12450	Remove deprecated methods from ReadOnlyWindowStore
Streams API	KAFKA-12880	Remove deprecated Count and SampledTotal in 3.0
Streams API	KAFKA-12451	Remove deprecation annotation on long-based read operations in WindowStore
Streams API	KAFKA-12568	Remove deprecated "KStream#groupBy/join", "Joined#named" overloads
Streams API	KAFKA-12849	Migrate TaskMetadata to interface with internal implementation
Streams API	KAFKA-7785	Remove PartitionGrouper interface and it's config and move DefaultPartitionGrouper to internal package
Streams API	KAFKA-7106	Remove segment/segmentInterval from Window definition

API or component	Issue link	Description
Streams API	KAFKA-8897	Increase Version of RocksDB
Streams API	KAFKA-12909	Allow users to opt-into spurious left/outer stream-stream join improvement
Tools	KAFKA-8405	Remove deprecated **kafka-preferred-replica-election** command
Tools	KAFKA-12588	Remove deprecated --zookeeper in shell commands

5.5.2. Mirror Maker 1.0 planned for removal in Kafka version 4.0

Kafka version 4.0 will be shipped in a future major release of AMQ Streams.

The following table shows a feature that will be deprecated in Kafka 3.0 and **removed** in Kafka 4.0.

Table 5.2. Components that will be deprecated in Kafka 3.0 and removed in Kafka 4.0

Component	Link to issue	Summary
Mirror Maker 1.0	KAFKA-12436	deprecate MirrorMaker v1

CHAPTER 6. FIXED ISSUES

The following sections list the issues fixed in AMQ Streams 1.8.x. Red Hat recommends that you upgrade to the latest patch release

For details of the issues fixed in Kafka 2.8.0, refer to the Kafka 2.8.0 Release Notes .

6.1. FIXED ISSUES FOR AMQ STREAMS 1.8.4

The AMQ Streams 1.8.4 patch release is now available.

The AMQ Streams product images have been upgraded to version 1.8.4.

For additional details about the issues resolved in AMQ Streams 1.8.4, see AMQ Streams 1.8.x Resolved Issues.

Log4j2 vulnerability

The 1.8.4 release fixes a remote code execution vulnerability for AMQ Streams components that use log4j2. The vulnerability could allow a remote code execution on the server if the system logs a string value from an unauthorized source. This affects log4j versions between 2.0 and 2.14.1.

For more information, see CVE-2021-44228.

6.2. FIXED ISSUES FOR AMQ STREAMS 1.8.0

Issue Number	Description
ENTMQST-1529	FileStreamSourceConnector stops when using a large file.
ENTMQST-2359	Kafka Bridge does not handle assignment and subscription.
ENTMQST-2453	The **kafka-exporter** pod restarts for no reason.
ENTMQST-2459	Running Kafka Exporter leads to high CPU usage.
ENTMQST-2511	Fine tune the health checks to stop Kafka Exporter restarting during rolling updates.
ENTMQST-2777	ENTMQST-2777 Custom Bridge labels are not set when the service template is not specified.
ENTMQST-2974	Changing the log level for Kafka Connect connectors only works temporarily.

Table 6.1. Fixed common vulnerabilities and exposures (CVEs)

Issue Number	Description
ENTMQST-1934	CVE-2020-9488 log4j: improper validation of certificate with host mismatch in SMTP appender [amq-st-1].
ENTMQST-2613	CVE-2020-13949 libthrift: potential DoS when processing untrusted payloads [amq-st-1].
ENTMQST-2617	CVE-2021-21290 netty: Information disclosure via the local system temporary directory [amq-st-1].
ENTMQST-2647	CVE-2021-21295 netty: possible request smuggling in HTTP/2 due missing validation [amq-st-1].
ENTMQST-2663	CVE-2021-27568 json-smart: uncaught exception may lead to crash or information disclosure [amq-st-1].
ENTMQST-2711	ENTMQST-2711 CVE-2021-21409 netty: Request smuggling via content-length header [amq-st-1].
ENTMQST-2821	CVE-2021-28168 jersey-common: jersey: Local information disclosure via system temporary directory [amq-st-1].
ENTMQST-2867	CVE-2021-29425 commons-io: apache-commons-io: Limited path traversal in Apache Commons IO 2.2 to 2.6 [amq-st-1].
ENTMQST-2908	ENTMQST-2908 CVE-2021-28165 jetty-server: jetty: Resource exhaustion when receiving an invalid large TLS frame [amq-st-1].
ENTMQST-2909	CVE-2021-28164 jetty-server: jetty: Ambiguous paths can access WEB-INF [amq-st-1].
ENTMQST-2910	CVE-2021-28163 jetty-server: jetty: Symlink directory exposes webapp directory contents [amq-st-1].
ENTMQST-2980	CVE-2021-28169 jetty-server: jetty: requests to the ConcatServlet and WelcomeFilter are able to access protected resources within the WEB-INF directory [amq-st-1].
ENTMQST-3023	CVE-2021-34428 jetty-server: jetty: SessionListener can prevent a session from being invalidated breaking logout [amq-st-1].

CHAPTER 7. KNOWN ISSUES

This section lists the known issues for AMQ Streams 1.8.

7.1. SMTP APPENDER FOR LOG4J

AMQ Streams ships with a potentially vulnerable version of log4j (**log4j-1.2.17.redhat-3**). The vulnerability lies with the SMTP appender functionality, which is not used by AMQ Streams in its default configuration.

Table 7.1. CVE issue

Issue Number	Description
ENTMQST-1934	CVE-2020-9488 log4j: improper validation of certificate with host mismatch in SMTP appender [amq-st-1].

Workaround

If you are using the SMTP appender, ensure that **mail.smtp.ssl.checkserveridentity** is set to **true**.

7.2. AMQ STREAMS CLUSTER OPERATOR ON IPV6 CLUSTERS

The AMQ Streams Cluster Operator does not start on Internet Protocol version 6 (IPv6) clusters.

Workaround

There are two workarounds for this issue.

Workaround one: Set the KUBERNETES_MASTER environment variable

1. Display the address of the Kubernetes master node of your OpenShift Container Platform cluster:

```
oc cluster-info
Kubernetes master is running at MASTER-ADDRESS
# ...
```

   Copy the address of the master node.

2. List all Operator subscriptions:

```
oc get subs -n OPERATOR-NAMESPACE
```

3. Edit the **Subscription** resource for AMQ Streams:

```
oc edit sub amq-streams -n OPERATOR_NAMESPACE
```

4. In **spec.config.env**, add the **KUBERNETES_MASTER** environment variable, set to the address of the Kubernetes master node. For example:

```
apiVersion: operators.coreos.com/v1alpha1
```

```
kind: Subscription
metadata:
  name: amq-streams
  namespace: OPERATOR-NAMESPACE
spec:
  channel: amq-streams-1.8.x
  installPlanApproval: Automatic
  name: amq-streams
  source: mirror-amq-streams
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: KUBERNETES_MASTER
      value: MASTER-ADDRESS
```

5. Save and exit the editor.
6. Check that the **Subscription** was updated:

```
oc get sub amq-streams -n OPERATOR-NAMESPACE
```

7. Check that the Cluster Operator **Deployment** was updated to use the new environment variable:

```
oc get deployment CLUSTER-OPERATOR-DEPLOYMENT-NAME
```

Workaround two: Disable hostname verification

1. List all Operator subscriptions:

```
oc get subs -n OPERATOR-NAMESPACE
```

2. Edit the **Subscription** resource for AMQ Streams:

```
oc edit sub amq-streams -n OPERATOR_NAMESPACE
```

3. In **spec.config.env**, add the **KUBERNETES_DISABLE_HOSTNAME_VERIFICATION** environment variable, set to **true**. For example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: amq-streams
  namespace: OPERATOR-NAMESPACE
spec:
  channel: amq-streams-1.8.x
  installPlanApproval: Automatic
  name: amq-streams
  source: mirror-amq-streams
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: KUBERNETES_DISABLE_HOSTNAME_VERIFICATION
      value: "true"
```

4. Save and exit the editor.
5. Check that the **Subscription** was updated:

```
oc get sub amq-streams -n OPERATOR-NAMESPACE
```

6. Check that the Cluster Operator **Deployment** was updated to use the new environment variable:

```
oc get deployment CLUSTER-OPERATOR-DEPLOYMENT-NAME
```

CHAPTER 8. SUPPORTED INTEGRATION PRODUCTS

AMQ Streams 1.8 supports integration with the following Red Hat products.

Red Hat Single Sign-On 7.4 and later

Provides OAuth 2.0 authentication and OAuth 2.0 authorization.

Red Hat 3scale API Management 2.6 and later

Secures the Kafka Bridge and provides additional API management features.

Red Hat Debezium 1.5

Monitors databases and creates event streams.

Red Hat Service Registry 2.0

Provides a centralized store of service schemas for data streaming.

For information on the functionality these products can introduce to your AMQ Streams deployment, refer to the AMQ Streams 1.8 documentation.

Additional resources

- Red Hat Single Sign-On Supported Configurations
- Red Hat 3scale API Management Supported Configurations
- Red Hat Debezium Supported Configurations
- Red Hat Service Registry Supported Configurations

CHAPTER 9. IMPORTANT LINKS

- Red Hat AMQ 7 Supported Configurations
- Red Hat AMQ 7 Component Details

Revised on 2021-12-14 20:09:28 UTC