Red Hat AMQ 2020.Q4

Release Notes for AMQ Streams 1.6 on RHEL

For use with AMQ Streams on Red Hat Enterprise Linux

Last Updated: 2022-02-01

Red Hat AMQ 2020.Q4 Release Notes for AMQ Streams 1.6 on RHEL

For use with AMQ Streams on Red Hat Enterprise Linux

Legal Notice

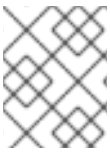
Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Streams 1.6 release.

Table of Contents

CHAPTER 1. FEATURES

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. KAFKA SUPPORT IN AMQ STREAMS 1.6.X (LONG TERM SUPPORT)

This section describes the versions of Kafka and ZooKeeper that are supported in AMQ Streams 1.6 and the subsequent patch releases.

AMQ Streams 1.6.x is the Long Term Support release for use with RHEL 7 and 8.

For information on support dates for AMQ LTS versions, see the Red Hat Knowledgebase solution How long are AMQ LTS releases supported?.

Only Kafka distributions built by Red Hat are supported. Previous versions of Kafka are supported in AMQ Streams 1.6.x only for upgrade purposes.

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

1.1.1. Kafka support in AMQ Streams 1.6.6 and 1.6.7

The AMQ Streams 1.6.6 and 1.6.7 releases support Apache Kafka version 2.6.3.

For upgrade instructions, see AMQ Streams and Kafka upgrades .

Refer to the Kafka 2.6.3 Release Notes for additional information.

Kafka 2.6.3 requires ZooKeeper version 3.5.9. Therefore, you do *not* need to upgrade ZooKeeper when upgrading from AMQ Streams 1.6.4 / 1.6.5.

1.1.2. Kafka support in AMQ Streams 1.6.4 and 1.6.5

The AMQ Streams 1.6.4 and 1.6.5 releases support and use Apache Kafka version 2.6.2 and ZooKeeper version 3.5.9.

For upgrade instructions, see AMQ Streams and Kafka upgrades .

Refer to the Kafka 2.6.2 Release Notes for additional information.

Kafka 2.6.2 requires ZooKeeper version 3.5.9. Therefore, you need to upgrade ZooKeeper when upgrading from AMQ Streams 1.6.0.

1.1.3. Kafka support in AMQ Streams 1.6.0

AMQ Streams 1.6.0 supports and uses Apache Kafka version 2.6.0.

For upgrade instructions, see AMQ Streams and Kafka upgrades .

Refer to the Kafka 2.5.0 and Kafka 2.6.0 Release Notes for additional information.

NOTE

Kafka 2.5.x is supported in AMQ Streams 1.6.0 only for upgrade purposes.

Kafka 2.6.0 requires the same ZooKeeper version as Kafka 2.5.x (ZooKeeper version 3.5.7 / 3.5.8). Therefore, you do *not* need to upgrade ZooKeeper when upgrading from AMQ Streams 1.5.

1.2. OAUTH 2.0 AUTHORIZATION

Support for OAuth 2.0 authorization moves out of Technology Preview to a generally available component of AMQ Streams.

If you are using OAuth 2.0 for token-based authentication, you can now also use OAuth 2.0 authorization rules to constrain client access to Kafka brokers.

AMQ Streams supports the use of OAuth 2.0 token-based authorization through Red Hat Single Sign-On Authorization Services, which allows you to manage security policies and permissions centrally.

Security policies and permissions defined in Red Hat Single Sign-On are used to grant access to resources on Kafka brokers. Users and clients are matched against policies that permit access to perform specific actions on Kafka brokers.

See Using OAuth 2.0 token-based authorization .

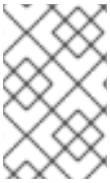
1.3. OPEN POLICY AGENT (OPA) INTEGRATION

Open Policy Agent (OPA) is an open-source policy engine. You can integrate OPA with AMQ Streams to act as a policy-based authorization mechanism for permitting client operations on Kafka brokers.

When a request is made from a client, OPA will evaluate the request against policies defined for Kafka access, then allow or deny the request.

You can define access control for Kafka clusters, consumer groups and topics. For instance, you can define an authorization policy that allows write access from a producer client to a specific broker topic.

See **KafkaAuthorizationOpa** schema reference



NOTE

- Red Hat does not support the OPA server.
- OPA integration is only supported on Open JDK 11.

CHAPTER 2. ENHANCEMENTS

The enhancements added in this release are outlined below.

2.1. KAFKA ENHANCEMENTS

For an overview of the enhancements introduced with:

- Kafka 2.6.2, refer to the Kafka 2.6.2 Release Notes (applies only to AMQ Streams 1.6.4)
- Kafka 2.6.1, refer to the Kafka 2.6.1 Release Notes (applies only to AMQ Streams 1.6.4)
- Kafka 2.6.0, refer to the Kafka 2.6.0 Release Notes

2.2. KAFKA BRIDGE ENHANCEMENTS

This release includes the following enhancements to the Kafka Bridge component of AMQ Streams.

Retrieve partitions and metadata

The Kafka Bridge now supports the following operations:

- Retrieve a list of partitions for a given topic:

```
GET /topics/{topicname}/partitions
```

- Retrieve metadata for a given partition, such as the partition ID, the leader broker, and the number of replicas:

```
GET /topics/{topicname}/partitions/{partitionid}
```

See the Kafka Bridge API reference.

Support for Kafka message headers

Messages sent using the Kafka Bridge can now include Kafka message headers.

In a POST request to the **/topics** endpoint, you can optionally specify headers in the message payload, which is contained in the request body. Message header values must be in binary format and encoded as Base64.

Example request with Kafka message header

```
curl -X POST \
  http://localhost:8080/topics/my-topic \
  -H 'content-type: application/vnd.kafka.json.v2+json' \
  -d '{
    "records": [
        {
            "key": "my-key",
            "value": "sales-lead-0001"
            "partition": 2
            "headers": [
              {
                "key": "key1",
```

```
            "value": "QXBhY2hlIEthZmthIGlzIHRoZSBib21iIQ=="
          }
        ]
      },
    ]
}'
```

See Requests to the Kafka Bridge .

2.3. MIRRORMAKER 2.0 TOPIC RENAMING UPDATE

The MirrorMaker 2.0 architecture supports bidirectional replication by automatically renaming remote topics to represent the source cluster. The name of the originating cluster is prepended to the name of the topic.

Optionally, you can now override automatic renaming by adding **IdentityReplicationPolicy** to the source connector configuration. With this configuration applied, topics retain their original names.

```
replication.policy.class= io.strimzi.kafka.connect.mirror.IdentityReplicationPolicy (1)
```

(1) Adds a policy that overrides the automatic renaming of remote topics. Instead of prepending the name with the name of the source cluster, the topic retains its original name.

The override is useful, for example, in an *active/passive* cluster configuration where you want to make backups or migrate data to another cluster. In either situation, you might not want automatic renaming of remote topics.

See Using AMQ Streams with MirrorMaker 2.0

2.4. OAUTH 2.0 AUTHENTICATION AND AUTHORIZATION

This release includes the following enhancements to OAuth 2.0 token-based authentication and authorization.

Session re-authentication

OAuth 2.0 authentication in AMQ Streams now supports *session re-authentication* for Kafka brokers. This defines the maximum duration of an authenticated OAuth 2.0 session between a Kafka client and a Kafka broker. Session re-authentication is supported for both types of token validation: fast local JWT and introspection endpoint.

You configure session re-authentication in the OAuth 2.0 configuration for Kafka brokers, in the **server.properties** file.

- To apply to all listeners, set the **connections.max.reauth.ms** property in milliseconds.
- To apply to a specific listener, set the **listener.name.*LISTENER-NAME*.oauthbearer.connections.max.reauth.ms** property in milliseconds. *LISTENER-NAME* is the case-insensitive name of the listener.

An authenticated session is closed if it exceeds the configured maximum session re-authentication time, or if the access token expiry time is reached. Then, the client must log in to the authorization server again, obtain a new access token, and then re-authenticate to the Kafka broker. This will establish a new authenticated session over the existing connection.

When re-authentication is next required, any operation that is attempted by the client (apart from re-authentication) will cause the broker to terminate the connection.

Example listener configuration for session re-authentication after 6 minutes

```
sasl.enabled.mechanisms=OAUTHBEARER
listeners=CLIENT://0.0.0.0:9092
# ...
listener.name.client.oauthbearer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \
  oauth.valid.issuer.uri="https://AUTH-SERVER-ADDRESS" \
  oauth.jwks.endpoint.uri="https://AUTH-SERVER-ADDRESS/jwks" \
  oauth.username.claim="preferred_username"  \
  oauth.client.id="kafka-broker" \
  oauth.client.secret="kafka-secret" \
  oauth.token.endpoint.uri="https://AUTH-SERVER-ADDRESS/token" ;
listener.name.client.oauthbearer.sasl.login.callback.handler.class=io.strimzi.kafka.oauth.client.JaasClientOauthLoginCallbackHandler
listener.name.client.oauthbearer.connections.max.reauth.ms=3600000
```

See: Session re-authentication for Kafka brokers and Configuring OAuth 2.0 support for Kafka brokers .

JWKS keys refresh interval

When configuring Kafka brokers to use fast local JWT token validation, you can now set the **oauth.jwks.refresh.min.pause.seconds** option in the listener configuration (in the **server.properties** file). This defines the minimum interval between attempts by the broker to refresh JSON Web Key Set (JWKS) public keys issued by the authorization server.

With this release, if the Kafka broker detects an unknown signing key, it attempts to refresh JWKS keys immediately and ignores the regular refresh schedule.

Example configuration for a 2-minute pause between attempts to refresh JWKS keys

```
listener.name.client.oauthbearer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \
  oauth.valid.issuer.uri="https://AUTH-SERVER-ADDRESS" \
  oauth.jwks.endpoint.uri="https://AUTH-SERVER-ADDRESS/jwks" \
  oauth.jwks.refresh.seconds="300" \
  oauth.jwks.refresh.min.pause.seconds="120" \
  # ...
  oauth.ssl.truststore.type="PKCS12" ;
```

The refresh schedule for JWKS keys is set in the **oauth.jwks.refresh.seconds** option. When an unknown signing key is encountered, a JWKS keys refresh is scheduled outside of the refresh schedule. The refresh will not start until the time since the last refresh reaches the interval specified in **oauth.jwks.refresh.min.pause.seconds**. The default value is **1**.

See Configuring OAuth 2.0 support for Kafka brokers .

Refreshing grants from Red Hat Single Sign-On

New configuration options have been added for OAuth 2.0 token-based authorization through Red Hat Single Sign-On. When configuring Kafka brokers, you can now define the following options related to refreshing grants from Red Hat SSO Authorization Services:

- **strimzi.authorization.grants.refresh.period.seconds**: The time between two consecutive grants refresh runs. The default value is **60**. If set to **0** or less, refreshing of grants is disabled.
- **strimzi.authorization.grants.refresh.pool.size**: The number of threads that can fetch grants for the active session in parallel. The default value is **5**.

See Using OAuth 2.0 token-based authorization and Configuring OAuth 2.0 authorization support

Detection of permission changes in Red Hat Single Sign-On

With this release, the **KeycloakRBACAuthorizer** (Red Hat SSO) authorization regularly checks for changes in permissions for the active sessions. Central user and permissions management changes are now detected in real time.

2.5. DEPRECATION OF ZOOKEEPER OPTION IN KAFKA ADMINISTRATIVE TOOLS

The **--zookeeper** option was deprecated in the following Kafka administrative tools:

- **bin/kafka-configs.sh**
- **bin/kafka-leader-election.sh**
- **bin/kafka-topics.sh**

When using these tools, you should now use the **--bootstrap-server** option to specify the Kafka broker to connect to. For example:

```
/bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

Although the **--zookeeper** option still works, it will be removed from all the administrative tools in a future Kafka release. This is part of ongoing work in the Apache Kafka project to remove Kafka's dependency on ZooKeeper.

The Using AMQ Streams on RHEL guide has been updated to use the **--bootstrap-server** option in several procedures.

CHAPTER 3. 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.

3.1. CLUSTER REBALANCING WITH CRUISE CONTROL

You can now install Cruise Control and use it to rebalance a Kafka cluster. Cruise Control helps to reduce the time and effort involved in running an efficient and balanced Kafka cluster.

A zipped distribution of Cruise Control is available for download from the Customer Portal. To install Cruise Control, you configure each Kafka broker to use the provided Metrics Reporter. Then, you set Cruise Control properties, including optimization goals, and start Cruise Control using the provided script.

The Cruise Control server is hosted on a single machine for the whole Kafka cluster.

When Cruise Control is running, you can use the REST API to:

- Generate dry run optimization proposals from multiple optimization goals
- Initiate an optimization proposal to rebalance the Kafka cluster

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.

CHAPTER 4. DEPRECATED FEATURES

There are no deprecated features for AMQ Streams 1.6.

CHAPTER 5. FIXED ISSUES

The following sections list the issues fixed in AMQ Streams 1.6.x. Red Hat recommends that you upgrade to the latest patch release if you are using AMQ Streams 1.6.x with RHEL 7 and 8.

For details of the issues fixed in:

- Kafka 2.6.3, refer to the Kafka 2.6.3 Release Notes
- Kafka 2.6.2, refer to the Kafka 2.6.2 Release Notes
- Kafka 2.6.1, refer to the Kafka 2.6.1 Release Notes
- Kafka 2.6.0, refer to the Kafka 2.6.0 Release Notes

5.1. FIXED ISSUES FOR AMQ STREAMS 1.6.7

The AMQ Streams 1.6.7 patch release (Long Term Support) is now available.

AMQ Streams 1.6.7 is the latest Long Term Support release for use with RHEL 7 and 8.

For additional details about the issues resolved in AMQ Streams 1.6.7, see AMQ Streams 1.6.x Resolved Issues.

Log4j vulnerabilities

AMQ Streams includes log4j 1.2.17. The release fixes a number of log4j vulnerabilities.

For more information on the vulnerabilities addressed in this release, see the following CVE articles:

- CVE-2022-23307
- CVE-2022-23305
- CVE-2022-23302
- CVE-2021-4104
- CVE-2020-9488
- CVE-2019-17571
- CVE-2017-5645

5.2. FIXED ISSUES FOR AMQ STREAMS 1.6.6

For additional details about the issues resolved in AMQ Streams 1.6.6, see AMQ Streams 1.6.x Resolved Issues.

Log4j2 vulnerabilities

AMQ Streams includes log4j2 2.17.1. The release fixes a number of log4j2 vulnerabilities.

For more information on the vulnerabilities addressed in this release, see the following CVE articles:

- CVE-2021-45046

- CVE-2021-45105
- CVE-2021-44832
- CVE-2021-44228

5.3. FIXED ISSUES FOR AMQ STREAMS 1.6.5

For additional details about the issues resolved in AMQ Streams 1.6.5, see AMQ Streams 1.6.x Resolved Issues.

Log4j2 vulnerability

The 1.6.5 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.

5.4. FIXED ISSUES FOR AMQ STREAMS 1.6.4

For additional details about the issues resolved in AMQ Streams 1.6.4, see AMQ Streams 1.6.x Resolved Issues.

5.5. FIXED ISSUES FOR AMQ STREAMS 1.6.0

Issue Number	Description
ENTMQST-2049	Kafka Bridge: Kafka consumer should be tracked with group-consumerid key
ENTMQST-2084	Zookeeper version on the docs doesn't match with the version in AMQ Streams 1.5

CHAPTER 6. KNOWN ISSUES

This section lists the known issues for AMQ Streams 1.6.

Issue Number

ENTMQST-2030 - kafka-ack reports **javax.management.InstanceAlreadyExistsException: kafka.admin.client:type=app-info,id=<client_id>with client.id set**

Description

If the **bin/kafka-acls.sh** utility is used in combination with the **--bootstrap-server** parameter to add or remove an ACL, the operation is successful but a warning is generated. The reason for the warning is that a second **AdminClient** instance is created. This will be fixed in a future release of Kafka.

CHAPTER 7. SUPPORTED INTEGRATION PRODUCTS

AMQ Streams 1.6 supports integration with the following Red Hat products.

- **Red Hat Single Sign-On 7.4 and later** for OAuth 2.0 authentication and OAuth 2.0 authorization
- **Red Hat Debezium 1.0 and later** for monitoring databases and creating event streams

For information on the functionality these products can introduce to your AMQ Streams deployment, refer to the AMQ Streams 1.6 documentation.

CHAPTER 8. IMPORTANT LINKS

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

Revised on 2022-02-01 16:35:47 UTC