Red Hat Integration 2022.Q1

Release Notes for Red Hat Integration 2022.Q1

What's new in Red Hat Integration

Last Updated: 2022-04-20
What's new in Red Hat Integration
Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license (“CC-BY-SA”). An explanation of CC-BY-SA is available at http://creativecommons.org/licenses/by-sa/3.0/. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation’s permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Describes the Red Hat Integration platform and provides the latest details on what’s new in this release.
# Table of Contents

CHAPTER 1. RED HAT INTEGRATION ................................................................. 4

CHAPTER 2. CAMEL EXTENSIONS FOR QUARKUS RELEASE NOTES ................................................................. 5
  2.1. CAMEL EXTENSIONS FOR QUARKUS FEATURES ................................................................. 5
  2.2. SUPPORTED PLATFORMS, CONFIGURATIONS, DATABASES, AND EXTENSIONS ................................................................. 5
  2.3. TECHNOLOGY PREVIEW EXTENSIONS ................................................................. 5
  2.4. KNOWN ISSUES ................................................................. 5
  2.5. IMPORTANT NOTES ................................................................. 6
  2.6. RESOLVED ISSUES ................................................................. 7
  2.7. DEPRECATED CAMEL EXTENSIONS FOR QUARKUS FEATURES ................................................................. 8
  2.8. ADDITIONAL RESOURCES ................................................................. 8

CHAPTER 3. DEBEZIUM RELEASE NOTES .............................................................................. 9
  3.1. DEBEZIUM DATABASE CONNECTORS ................................................................. 9
  3.2. DEBEZIUM SUPPORTED CONFIGURATIONS ................................................................. 10
  3.3. DEBEZIUM INSTALLATION OPTIONS ................................................................. 10
  3.4. NEW DEBEZIUM FEATURES ................................................................. 10
  3.5. DEPRECATED DEBEZIUM FEATURES ................................................................. 12

CHAPTER 4. CAMEL K RELEASE NOTES .............................................................................. 13
  4.1. NEW CAMEL K FEATURES ................................................................. 13
  4.2. SUPPORTED CONFIGURATIONS ................................................................. 13
   4.2.1. Camel K Operator metadata ................................................................. 13
  4.3. IMPORTANT NOTES ................................................................. 14
  4.4. SUPPORTED CAMEL QUARKUS EXTENSIONS ................................................................. 14
   4.4.1. Supported Camel Quarkus connector extensions ................................................................. 15
   4.4.2. Supported Camel Quarkus dataformat extensions ................................................................. 15
   4.4.3. Supported Camel Quarkus language extensions ................................................................. 16
   4.4.4. Supported Camel K traits ................................................................. 16
  4.5. SUPPORTED KAMELETS ................................................................. 17
  4.6. CAMEL K KNOWN ISSUES ................................................................. 20
  4.7. CAMEL K FIXED ISSUES ................................................................. 21
   4.7.1. Enhancements in Camel K 1.6.4 ................................................................. 21
   4.7.2. Bugs resolved in Camel K 1.6.4 ................................................................. 22

CHAPTER 5. SERVICE REGISTRY RELEASE NOTES .............................................................................. 24
  5.1. SERVICE REGISTRY INSTALLATION OPTIONS ................................................................. 24
  5.2. SERVICE REGISTRY PLATFORM COMPONENT VERSIONS ................................................................. 24
  5.3. SERVICE REGISTRY NEW FEATURES ................................................................. 24
   Service Registry security ................................................................. 24
   Service Registry core ................................................................. 25
   Service Registry data storage ................................................................. 25
   Service Registry v2 REST API ................................................................. 25
   Service Registry Operator ................................................................. 25
   Service Registry user documentation and examples ................................................................. 26
  5.4. SERVICE REGISTRY DEPRECATED AND REMOVED FEATURES ................................................................. 26
   Service Registry deprecated features ................................................................. 26
   Service Registry removed features ................................................................. 26
  5.5. MIGRATING SERVICE REGISTRY DEPLOYMENTS ................................................................. 26
  5.6. SERVICE REGISTRY RESOLVED ISSUES ................................................................. 26
   Service Registry core resolved issues ................................................................. 26
   Service Registry Operator resolved issues ................................................................. 28
5.7. SERVICE REGISTRY KNOWN ISSUES
Service Registry core known issues
Service Registry operator known issues

CHAPTER 6. RED HAT INTEGRATION OPERATORS
6.1. WHAT OPERATORS ARE
6.2. RED HAT INTEGRATION COMPONENT OPERATORS
  6.2.1. 3scale Operators
  6.2.2. AMQ Operators
  6.2.3. Camel K Operator
  6.2.4. Fuse Operators
  6.2.5. Service Registry Operator
6.3. RED HAT INTEGRATION OPERATOR (DEPRECATED)
  6.3.1. Supported components
  6.3.2. Support life cycle
  6.3.3. Fixed issues
CHAPTER 1. RED HAT INTEGRATION

Red Hat Integration is a comprehensive set of integration and event processing technologies for creating, extending, and deploying container-based integration services across hybrid and multicloud environments. Red Hat Integration provides an agile, distributed, and API-centric solution that organizations can use to connect and share data between applications and systems required in a digital world.

Red Hat Integration includes the following capabilities:

- Real-time messaging
- Cross-datacenter message streaming
- API connectivity
- Application connectors
- Enterprise integration patterns
- API management
- Data transformation
- Service composition and orchestration

Additional resources

- Understanding enterprise integration
CHAPTER 2. CAMEL EXTENSIONS FOR QUARKUS RELEASE NOTES

2.1. CAMEL EXTENSIONS FOR QUARKUS FEATURES

Fast startup and low RSS memory
Using the optimized build-time and ahead-of-time (AOT) compilation features of Quarkus, your Camel application can be pre-configured at build time resulting in fast startup times.

Application generator
Use the Quarkus application generator to bootstrap your application and discover its extension ecosystem.

Highly configurable
All of the important aspects of a Camel Extensions for Quarkus application can be set up programmatically with CDI (Contexts and Dependency Injection) or via configuration properties. By default, a CamelContext is configured and automatically started for you. Check out the Configuring your Quarkus applications guide for more information on the different ways to bootstrap and configure an application.

Integrates with existing Quarkus extensions
Camel Extensions for Quarkus provides extensions for libraries and frameworks that are used by some Camel components which inherit native support and configuration options.

2.2. SUPPORTED PLATFORMS, CONFIGURATIONS, DATABASES, AND EXTENSIONS

- For information about supported platforms, configurations, and databases in Camel Extensions for Quarkus version 2.2, see the Supported Configuration page on the Customer Portal (login required).

- For a list of Red Hat Camel Extensions for Quarkus extensions and the Red Hat support level for each extension, see the Extensions Overview chapter of the Camel Extensions for Quarkus Reference (login required).

2.3. TECHNOLOGY PREVIEW EXTENSIONS

Red Hat does not provide support for Technology Preview components provided with this release of Camel Extensions for Quarkus. Items designated as Technology Preview in the Extensions Overview chapter of the Camel Extensions for Quarkus Reference have limited supportability, as defined by the Technology Preview Features Support Scope.

2.4. KNOWN ISSUES

CAMEL-17158 AWS2 SQS When sending messages to a queue that has delay, the delay is not respected
If you create a queue with a delay, the messages sent using the camel-aws2-sqs component as a producer do not respect the delay that has been set for the queue.
The reason for this behavior is that Camel sets '0s' as the default delay when sending messages that override the queue settings.
As a workaround, you should set the same delay settings when using the Camel producer. For example, if you create a queue with a 5s delay, you should also set a 5s delay when using the `camel-aws2-sqs` producer.

**ENTESB-17763 Missing productised transitive deps of camel-quarkus-jira extensions**

Applications using the `camel-quarkus-jira` extension require an additional Maven repository `https://packages.atlassian.com/maven-external/` to be configured either in the Maven settings.xml file or in the pom.xml file of the application project.

**ENTESB-18306 Missing productized netty-transport-native-epoll:jar:linux-aarch_64 in NSQ, HDFS and Spark.**

The reason for this behavior is that the native epoll libraries are not included in the `camel-quarkus-2.2.1-product` build. However, as these libraries are not required for NSQ, HDFS or Spark components, the recommended workaround is to exclude `netty-all` from your applications and include `quarkus-netty` as a dependency.

For example, you can update your application’s pom.xml for the hdfs component as follows:

```xml
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-hdfs</artifactId>
  <exclusions>
    <exclusion>
      <groupId>io.netty</groupId>
      <artifactId>netty-all</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-netty</artifactId>
</dependency>
```

### 2.5. IMPORTANT NOTES

**Camel upgraded from version 3.11.1 to version 3.11.5**

Camel Extensions for Quarkus version 2.2.1 has been upgraded from Camel version 3.11.1 to Camel version 3.11.5. For additional information about each intervening Camel patch release, please see the following:

- Apache Camel 3.11.2 Release Notes
- Apache Camel 3.11.3 Release Notes
- Apache Camel 3.11.4 Release Notes
- Apache Camel 3.11.5 Release Notes

**CVE-2021-44228 log4j-core: Remote code execution in Log4j 2.x**

In Camel Extensions for Quarkus version 2.2.1, the following artifacts are no longer managed as the Camel Extensions for Quarkus extensions do not depend on these artifacts:

- org.apache.logging.log4j:log4j-1.2-api
- org.apache.logging.log4j:log4j-core
• org.apache.logging.log4j:log4j-jcl
• org.apache.logging.log4j:log4j-jul
• org.apache.logging.log4j:log4j-slf4j-impl
• org.apache.logging.log4j:log4j-web

If your application adds a dependency for any of these artifacts, please make sure to use the latest version of Log4j 2.x to avoid any known CVEs related to versions before Log4j 2.17.1

NOTE

Change of minimum required Apache Maven version to 3.8.1

In this release of Camel Extensions for Quarkus version 2.2.1, the minimum version of Apache Maven required for compiling Red Hat build of Quarkus projects changes to 3.8.1. You must upgrade your installation of Apache Maven to version 3.8.1 to be able to compile projects based on Red Hat build of Quarkus 2.2. This upgrade is required because it addresses a security issue that can potentially make your Apache Maven builds vulnerable to man-in-the-middle attacks. For more information about this security vulnerability, see the entry about CVE-2021-26291 on the Red Hat Customer Portal.

2.6. RESOLVED ISSUES

ENTESB-18560 JSONPath output in Native mode different from JVM mode

Registration for reflection of a class used by the JSONPath language was missing. In previous versions of Camel Extensions for Quarkus, the output of JSONPath expressions in Native mode was as follows:

```java
{name="Jan", age=28}
```

In Camel Extensions for Quarkus version 2.2.1, this issue has been resolved and the output of JSONPath expressions in Native mode is now as follows:

```java
{"name":"Jan","age":28}
```

ENTESB-18016 Quarkus Dev UI referring to community documentation instead of Red Hat product documentation

When running a project in Quarqu dev mode using mvn quarkus:dev, Quarkus provides the Dev UI via the endpoint /q/dev. This interface displays deployed extensions containing links to the relevant documentation pages. In previous versions of Camel Extensions for Quarkus, these links pointed to community extension pages. In this release, these links have been updated to point to Red Hat product documentation pages.

ENTESB-17855 geronimo-jms_*_spec* artifacts replaced by jakarta.jms artifacts

In this release, the org.apache.geronimo.specs:geronimo-jms_1.1_spec and org.apache.geronimo.specs:geronimo-jms_2.0_spec artifacts used in various JMS related extensions have been replaced by the newer and vendor neutral jakarta.jms:jakarta.jms-api equivalent.

ENTESB-17939 Replace javax.activation in favor of jakarta.activation

In this release, the com.sun.activation:javax.activation and javax.activation:activation artifacts {name=Jan, age=28}

{"name":"Jan","age":28}
In this release, the `com.sun.activation:javax.activation` and `javax.activation:activation` artifacts used in various Camel Extensions for Quarkus extensions were replaced by the newer `com.sun.activation:jakarta.activation` equivalent.

### 2.7. DEPRECATED CAMEL EXTENSIONS FOR QUARKUS FEATURES

**Elasticsearch Rest extension**

The `camel-quarkus-elasticsearch-rest` extension for Camel Extensions for Quarkus is deprecated in this release and scheduled for removal in a future release.

### 2.8. ADDITIONAL RESOURCES

- Supported Configurations
- Camel Extensions for Quarkus
- Getting Started with Camel Extensions for Quarkus
- Developing Applications with Camel Extensions for Quarkus
CHAPTER 3. DEBEZIUM RELEASE NOTES

Debezium is a distributed change data capture platform that captures row-level changes that occur in database tables and then passes corresponding change event records to Apache Kafka topics. Applications can read these change event streams and access the change events in the order in which they occurred. Debezium is built on Apache Kafka and is deployed and integrated with AMQ Streams.

The following topics provide release details:

- Section 3.1, “Debezium database connectors”
- Section 3.2, “Debezium supported configurations”
- Section 3.3, “Debezium installation options”
- Section 3.4, “New Debezium features”
- Section 3.5, “Deprecated Debezium features”

3.1. DEBEZIUM DATABASE CONNECTORS

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

- Db2
- MongoDB
- MySQL
- Oracle (Technology Preview)
- PostgreSQL
- SQL Server

**NOTE**

- The Db2 connector requires the use of the abstract syntax notation (ASN) libraries, which are available as a standard part of Db2 for Linux.
  - To use the ASN libraries, you must have a license for IBM InfoSphere Data Replication (IIDR).
  - You do not have to install IIDR to use the libraries.
- Currently, you cannot use the transaction metadata feature of the Debezium MongoDB connector with MongoDB 4.2.
- The Debezium PostgreSQL connector requires you to use the *pgoutput* logical decoding output plug-in, which is the default for PostgreSQL versions 10 and later.
- To use the Debezium Oracle connector, you must download a copy of the Oracle JDBC driver (*ojdbc8.jar*) from Oracle.

Additional resources
3.2. DEBEZIUM SUPPORTED CONFIGURATIONS

For information about Debezium supported configurations, including information about supported database versions, see the Debezium 1.7 Supported configurations page.

**AMQ Streams new API version**

Debezium runs on AMQ Streams 2.0.

AMQ Streams now supports the v1beta2 API version, which updates the schemas of the AMQ Streams custom resources. Older API versions are deprecated. After you upgrade to AMQ Streams 1.7, but before you upgrade to AMQ Streams 1.8 or later, you must upgrade your custom resources to use API version v1beta2.

For more information, see the Debezium User Guide.

3.3. DEBEZIUM INSTALLATION OPTIONS

You can install Debezium with AMQ Streams on OpenShift or RHEL:

- Installing Debezium on OpenShift
- Installing Debezium on RHEL

3.4. NEW DEBEZIUM FEATURES

Debezium 1.7 includes the following updates:

**New deployment mechanism**

You can now use AMQ Streams to deploy Debezium connectors by using a new AMQ Streams build mechanism that is based on Maven artifacts. For more information, see Debezium documentation.

Debezium documentation

- Information about how to enable and use Debezium signaling tables to trigger ad hoc incremental snapshots: Sending signals to a Debezium connector.
- Revised deployment instructions in the Debezium User Guide.
  - Deploying Debezium Db2 connectors
  - Deploying Debezium MongoDB connectors
  - Deploying Debezium MySQL connectors
  - Deploying Debezium Oracle connectors
  - Deploying Debezium PostgreSQL connectors
  - Deploying Debezium SQL Server connectors

**Technology Preview features**
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 implementing any Technology Preview features in production environments. Technology Preview features provide 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.

Sending signals to a Debezium connector

The Integration signaling mechanism provides a way to modify the behavior of a connector, or to trigger the connector to perform a one-time action, such as initiating an ad hoc incremental snapshot of a table.

CloudEvents converter

Emits change event records that conform to the CloudEvents specification. Avro encoding type is now supported for the CloudEvents envelope structure.

Outbox event router

SMT that supports the outbox pattern for safely and reliably exchanging data between multiple (micro) services.

Debezium Oracle connector

Connector for Oracle Database. This release of the Debezium Oracle connector provides the following capabilities:

- Support tracking DDL changes for Oracle.
- Ability to perform snapshots without locking.
- Improved de-duplication checking for change event buffering.
- Improved SCN gap detection during streaming.
- Mining can be scoped to a specific pluggable database (PDB).
- Ability to skip or exclude redo entries by username.
- Stability improvements.
- An improved DML statement parser.
- Ability to capture changes from multiple schemas within the same database or pluggable-database.
- New performance-related JMX metrics.
- Ability to configure the precision of temporal values through the time.precision.mode property.
- Compatibility with environments that run multiple Archiver process (ARC) processes.
- Ability to process messages across multiple archive log destinations (works alongside Oracle Data Guard).
3.5. DEPRECATED DEBEZIUM FEATURES

MonitoredTables option for connector snapshot and streaming metrics

The MonitoredTables option for Debezium connector metrics is deprecated in this release and scheduled for removal in a future release. Use the CapturedTables metric in its place.
CHAPTER 4. CAMEL K RELEASE NOTES

Camel K is a lightweight integration framework built from Apache Camel K that runs natively in the cloud on OpenShift. Camel K is specifically designed for serverless and microservice architectures. You can use Camel K to instantly run integration code written in Camel Domain Specific Language (DSL) directly on OpenShift.

Using Camel K with OpenShift Serverless and Knative, containers are automatically created only as needed and are autoscaled under load up and down to zero. This removes the overhead of server provisioning and maintenance and enables you to focus instead on application development.

Using Camel K with OpenShift Serverless and Knative Eventing, you can manage how components in your system communicate in an event-driven architecture for serverless applications. This provides flexibility and creates efficiencies using a publish/subscribe or event-streaming model with decoupled relationships between event producers and consumers.

4.1. NEW CAMEL K FEATURES

The Camel K provides cloud-native integration with the following main features:

- Knative Serving for autoscaling and scale-to-zero
- Knative Eventing for event-driven architectures
- Performance optimizations using Quarkus runtime by default
- Camel integrations written in Java or YAML DSL
- Monitoring of integrations using Prometheus in OpenShift
- Quickstart tutorials
- Kamelet Catalog for connectors to external systems such as AWS, Jira, and Salesforce
- Support for Timer and Log Kamelets
- Metering for Camel K Operator and pods

4.2. SUPPORTED CONFIGURATIONS

For information about Camel K supported configurations, standards, and components, see the following Customer Portal articles:

- Camel K Supported Configurations
- Camel K Component Details

4.2.1. Camel K Operator metadata

The Camel K includes updated Operator metadata used to install Camel K from the OpenShift OperatorHub. This Operator metadata includes the Operator bundle format for release packaging, which is designed for use with OpenShift Container Platform 4.6 or later.

Additional resources
4.3. IMPORTANT NOTES

Important notes for the Red Hat Integration - Camel K release:

CVE-2022-22965 spring-framework: RCE via Data Binding on JDK 9+
A patched version of Camel K (version 1.6.5) has been released to address the spring-framework security issue, CVE-2022-22965. To update your Camel K deployments and application projects to pick up this patched version, follow the upgrade instructions in Chapter 4. Upgrading Camel K from the Getting Started with Camel K guide.

CVE-2021-44228 log4j-core: Remote code execution in Log4j 2.x
A patched version of Camel K (version 1.6.0-1) has been released to address the Log4j 2.x security issue, CVE-2021-44228 (popularly known as Log4Shell). To update your Camel K deployments and application projects to pick up this patched version, follow the upgrade instructions in Chapter 4. Upgrading Camel K from the Getting Started with Camel K guide. The patched version of Camel K is delivered through the latest Operator channel.

Supported Enterprise Integration Patterns (EIP) in Camel K
All Camel 3 EIP patterns, except the following, are fully supported for Camel K:

- Circuit Breaker
- Saga
- Change Data Capture

YAML DSL Limitations
YAML DSL integrations are supported in Camel K 1.6.5, but the error messaging for incorrect YAML DSL code is still in development.

JAVA DSL Limitations
Java DSL in Camel K 1.6.5 is limited to a single class/configure method and any utility must be provided in third party JARS. The endpoint URIs must be defined directly in the endpoint strings for the Camel K automatic dependency support, otherwise you must specify the dependencies in modeline.

XML DSL is not supported
XML DSL is not supported in Camel K 1.6.5.

Camel K 1.6.5 runtime can only access Maven repos that support HTTPS
You can only use the Maven repositories that are secured by HTTPS. The insecure HTTP protocol is no longer be supported.

4.4. SUPPORTED CAMEL QUARKUS EXTENSIONS

This section lists the Camel Quarkus extensions that are supported for this release of Camel K (only when used inside a Camel K application).

NOTE
These Camel Quarkus extensions are supported only when used inside a Camel K application. These Camel Quarkus extensions are not supported for use in standalone mode (without Camel K).
### 4.4.1. Supported Camel Quarkus connector extensions

The following table shows the Camel Quarkus connector extensions that are supported for this release of Camel K (only when used inside a Camel K application).

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS 2 Kinesis</td>
<td>camel-quarkus-aws2-kinesis</td>
</tr>
<tr>
<td>AWS 2 Lambda</td>
<td>camel-quarkus-aws2-lambda</td>
</tr>
<tr>
<td>AWS 2 S3 Storage Service</td>
<td>camel-quarkus-aws2-s3</td>
</tr>
<tr>
<td>AWS 2 Simple Notification System (SNS)</td>
<td>camel-quarkus-aws2-sns</td>
</tr>
<tr>
<td>AWS 2 Simple Queue Service (SQS)</td>
<td>camel-quarkus-aws2-sqs</td>
</tr>
<tr>
<td>File</td>
<td>camel-quarkus-file</td>
</tr>
<tr>
<td>FTP</td>
<td>camel-quarkus-ftp</td>
</tr>
<tr>
<td>FTPS</td>
<td>camel-quarkus-ftp</td>
</tr>
<tr>
<td>SFTP</td>
<td>camel-quarkus-ftp</td>
</tr>
<tr>
<td>HTTP</td>
<td>camel-quarkus-http</td>
</tr>
<tr>
<td>JMS</td>
<td>camel-quarkus-jms</td>
</tr>
<tr>
<td>Kafka</td>
<td>camel-quarkus-kafka</td>
</tr>
<tr>
<td>Kamelets</td>
<td>camel-quarkus-kamelet</td>
</tr>
<tr>
<td>Metrics</td>
<td>camel-quarkus-microprofile-metrics</td>
</tr>
<tr>
<td>MongoDB</td>
<td>camel-quarkus-mongodb</td>
</tr>
<tr>
<td>Salesforce</td>
<td>camel-quarkus-salesforce</td>
</tr>
<tr>
<td>SQL</td>
<td>camel-quarkus-sql</td>
</tr>
<tr>
<td>Timer</td>
<td>camel-quarkus-timer</td>
</tr>
</tbody>
</table>

### 4.4.2. Supported Camel Quarkus dataformat extensions

The following table shows the Camel Quarkus dataformat extensions that are supported for this release of Camel K (only when used inside a Camel K application).
### 4.4.3. Supported Camel Quarkus language extensions

In this release, Camel K supports the following Camel Quarkus language extensions (for use in Camel expressions and predicates):

- Constant
- ExchangeProperty
- File
- Header
- Ref
- Simple
- Tokenize
- JsonPath

### 4.4.4. Supported Camel K traits

In this release, Camel K supports the following Camel K traits:

- Builder trait
- Camel trait
- Container trait
- Dependencies trait
- Deployer trait
- Deployment trait
- Environment trait
- Jvm trait
- Kamelets trait
- Owner trait
4.5. SUPPORTED KAMELETS

The following table lists the kamelets that are provided as OpenShift resources when you install the Camel K operator.

For details about these kamelets, go to: https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.6

For information about how to use kamelets to connect applications and services, see https://access.redhat.com/documentation/en-us/red_hat_integration/2021.q4/html-single/integrating_applications_with_kamelets.

**IMPORTANT**

Kamelets marked with an asterisk (*) are Technology Preview features only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production.

These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

Table 4.1. Kamelets provided with the Camel K operator

<table>
<thead>
<tr>
<th>Kamelet</th>
<th>File name</th>
<th>Type (Sink, Source, Action)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avro Deserialize action</td>
<td>avro-deserialize-action.kamelet.yaml</td>
<td>Action (data conversion)</td>
</tr>
<tr>
<td>Avro Serialize action</td>
<td>avro-serialize-action.kamelet.yaml</td>
<td>Action (data conversion)</td>
</tr>
<tr>
<td>AWS 2 S3 sink</td>
<td>aws-s3-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>AWS 2 S3 source</td>
<td>aws-s3-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>Kamelet</td>
<td>File name</td>
<td>Type (Sink, Source, Action)</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>AWS 2 S3 Streaming Upload sink</td>
<td>aws-s3-streaming-upload-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>AWS 2 Kinesis sink</td>
<td>aws-kinesis-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>AWS 2 Kinesis source</td>
<td>aws-kinesis-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>AWS 2 Lambda sink</td>
<td>aws-lambda-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>AWS 2 Simple Notification System sink</td>
<td>aws-sns-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>AWS 2 Simple Queue Service sink</td>
<td>aws-sqs-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>AWS 2 Simple Queue Service source</td>
<td>aws-sqs-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>AWS SQS FIFO sink</td>
<td>aws-sqs-fifo-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Cassandra sink*</td>
<td>cassandra-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Cassandra source*</td>
<td>cassandra-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>Elasticsearch Index sink*</td>
<td>elasticsearch-index-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Extract Field action</td>
<td>extract-field-action.kamelet.yaml</td>
<td>Action</td>
</tr>
<tr>
<td>FTP sink</td>
<td>ftp-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>FTP source</td>
<td>ftp-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>Has Header Key Filter action</td>
<td>has-header-filter-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
<tr>
<td>Hoist Field action</td>
<td>hoist-field-action.kamelet.yaml</td>
<td>Action</td>
</tr>
<tr>
<td>HTTP sink</td>
<td>http-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Insert Field action</td>
<td>insert-field-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
<tr>
<td>Kamelet</td>
<td>File name</td>
<td>Type (Sink, Source, Action)</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Insert Header action</td>
<td>insert-header-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
<tr>
<td>Is Tombstone Filter action</td>
<td>is-tombstone-filter-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
<tr>
<td>Jira source*</td>
<td>jira-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>JMS sink</td>
<td>jms-amqp-10-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>JMS source</td>
<td>jms-amqp-10-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>JSON Deserialize action</td>
<td>json-deserialize-action.kamelet.yaml</td>
<td>Action (data conversion)</td>
</tr>
<tr>
<td>JSON Serialize action</td>
<td>json-serialize-action.kamelet.yaml</td>
<td>Action (data conversion)</td>
</tr>
<tr>
<td>Kafka sink</td>
<td>kafka-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Kafka source</td>
<td>kafka-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>Kafka Topic Name Filter action</td>
<td>topic-name-matches-filter-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
<tr>
<td>Log sink</td>
<td>log-sink.kamelet.yaml</td>
<td>Sink (for development and testing purposes)</td>
</tr>
<tr>
<td>Mask Fields action</td>
<td>mask-field-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
<tr>
<td>Message TimeStamp Router action</td>
<td>message-timestamp-router-action.kamelet.yaml</td>
<td>Action (router)</td>
</tr>
<tr>
<td>MongoDB sink</td>
<td>mongodb-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>MongoDB source</td>
<td>mongodb-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>MySQL sink</td>
<td>mysql-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Kamelet</td>
<td>File name</td>
<td>Type (Sink, Source, Action)</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>PostgreSQL sink</td>
<td>postgresql-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Predicate filter action</td>
<td>predicate-filter-action.kamelet.yaml</td>
<td>Action (router/filter)</td>
</tr>
<tr>
<td>Protobuf Deserialize action</td>
<td>protobuf-deserialize-action.kamelet.yaml</td>
<td>Action (data conversion)</td>
</tr>
<tr>
<td>Protobuf Serialize action</td>
<td>protobuf-serialize-action.kamelet.yaml</td>
<td>Action (data conversion)</td>
</tr>
<tr>
<td>Regex Router action</td>
<td>regex-router-action.kamelet.yaml</td>
<td>Action (router)</td>
</tr>
<tr>
<td>Replace Field action</td>
<td>replace-field-action.kamelet.yaml</td>
<td>Action</td>
</tr>
<tr>
<td>Salesforce source</td>
<td>salesforce-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>SFTP sink</td>
<td>sftp-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>SFTP source</td>
<td>sftp-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>Slack source</td>
<td>slack-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>SQL Server Database sink</td>
<td>sqlserver-sink.kamelet.yaml</td>
<td>Sink</td>
</tr>
<tr>
<td>Telegram source*</td>
<td>telegram-source.kamelet.yaml</td>
<td>Source</td>
</tr>
<tr>
<td>Timer source</td>
<td>timer-source.kamelet.yaml</td>
<td>Source (for development and testing purposes)</td>
</tr>
<tr>
<td>TimeStamp Router action</td>
<td>timestamp-router-action.kamelet.yaml</td>
<td>Action (router)</td>
</tr>
<tr>
<td>Value to Key action</td>
<td>value-to-key-action.kamelet.yaml</td>
<td>Action (data transformation)</td>
</tr>
</tbody>
</table>

### 4.6. CAMEL K KNOWN ISSUES

The following known issues apply to the Camel K 1.6.5:

**ENTESB-15306** - CRD conflicts between Camel K and Fuse Online
If an older version of Camel K has ever been installed in the same OpenShift cluster, installing Camel K from the OperatorHub fails due to conflicts with custom resource definitions. For example, this includes older versions of Camel K previously available in Fuse Online.

For a workaround, you can install Camel K in a different OpenShift cluster, or enter the following command before installing Camel K:

```bash
$ oc get crds -l app=camel-k -o json | oc delete -f -
```

**ENTESB-15858** - Added ability to package and run Camel integrations locally or as container images

Packaging and running Camel integrations locally or as container images is not currently included in the Camel K and has community-only support.

For more details, see the [Apache Camel K community](https://camel.apache.org/).

**ENTESB-16477** - Unable to download jira client dependency with productized build

When using Camel K operator, the integration is unable to find dependencies for jira client. The workaround is to add the atlassian repo manually.

```yaml
apiVersion: camel.apache.org/v1
description: IntegrationPlatform
metadata:
  labels:
    app: camel-k
    name: camel-k
spec:
  configuration:
    - type: repository
      value: <atlassian repo here>
```

**ENTESB-17033** - Camel-K ElasticsearchComponent options ignored

When configuring the Elasticsearch component, the Camel K ElasticsearchComponent options are ignored. The workaround is to add `getContext().setAutowiredEnabled(false)` when using the Elasticsearch component.

**ENTESB-17061** - Can’t run mongo-db-source kamelet route with non-admin user – Failed to start route mongodb-source-1 because of null

It is not possible to run `mongo-db-source kamelet` route with non-admin user credentials. Some part of the component require admin credentials hence it is not possible run the route as a non-admin user.

### 4.7. CAMEL K FIXED ISSUES

The following sections list the issues that have been fixed in Camel K 1.6.4.

- Section 4.7.1, “Enhancements in Camel K 1.6.4”
- Section 4.7.2, “Bugs resolved in Camel K 1.6.4”

### 4.7.1. Enhancements in Camel K 1.6.4

The following table lists the enhancements in Camel K 1.6.4.
### Table 4.2. Camel K 1.6.4 Enhancements

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTESB-14406</td>
<td>Provide metering labels for Camel K Operator and deployed pods</td>
</tr>
<tr>
<td>ENTESB-18389</td>
<td>Bom in Kamelets Catalog</td>
</tr>
<tr>
<td>ENTESB-18389</td>
<td>Kamelet Catalog to use <code>.spec.template</code> vs <code>.spec.flow</code></td>
</tr>
</tbody>
</table>

### 4.7.2. Bugs resolved in Camel K 1.6.4

The following table lists the resolved bugs in Camel K 1.6.4.

### Table 4.3. Camel K 1.6.4 Resolved Bugs

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTESB-14343</td>
<td>Respect cluster-wide proxy settings in Camel K</td>
</tr>
<tr>
<td>ENTESB-15385</td>
<td>CVE-2020-27218 jetty: buffer not correctly recycled in Gzip Request inflation [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-15448</td>
<td>CVE-2020-8908 guava: local information disclosure via temporary directory created with unsafe permissions [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-16081</td>
<td>CVE-2021-20293 resteasy-core: RESTEasy: PathParam in RESTEasy can lead to a reflected XSS attack [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-16165</td>
<td>CVE-2021-21349 xstream: SSRF can be activated unmarshalling with XStream to access data streams from an arbitrary URL referencing a resource in an intranet or the local host [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-16366</td>
<td>CVE-2021-28168 jersey-common: jersey: Local information disclosure via system temporary directory [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-16500</td>
<td>CVE-2021-26291 maven-core: maven: Block repositories using http by default [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-16521</td>
<td>CVE-2020-15522 bouncycastle: Timing issue within the EC math library [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-16629</td>
<td>CVE-2021-28170 jakarta.el: jakarta-el: ELParserTokenManager enables invalid EL expressions to be evaluate [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-17118</td>
<td>CVE-2021-33813 jdom: XXE allows attackers to cause a DoS via a crafted HTTP request [rhint-camel-k-1]</td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENTESB-17119</td>
<td>CVE-2021-33813 jdom2: jdom: XXE allows attackers to cause a DoS via a crafted HTTP request [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-17149</td>
<td>CVE-2021-3690 undertow: buffer leak on incoming websocket PONG message may lead to DoS [rhint-camel-k-1]</td>
</tr>
<tr>
<td>ENTESB-17407</td>
<td>[Camel K] Allow to specify HTTP Proxy settings</td>
</tr>
<tr>
<td>ENTESB-17594</td>
<td>Deserialize custom headers in Kafka Kamelet source</td>
</tr>
<tr>
<td>ENTESB-17851</td>
<td>Unable to produce knative event to custom broker</td>
</tr>
<tr>
<td>ENTESB-17957</td>
<td>Front-end &quot;FailedMount&quot; in quickstart Camel K: Event Streaming Example</td>
</tr>
<tr>
<td>ENTESB-17959</td>
<td>Jitpack dependency in quickstart refers to main–SNAPSHOT in 1.6.x branch</td>
</tr>
<tr>
<td>ENTESB-18003</td>
<td>Remove flawed library from mrcc repo for Camel K</td>
</tr>
<tr>
<td>ENTESB-18296</td>
<td>kamel binaries present in the Camel-K MRRC</td>
</tr>
<tr>
<td>ENTESB-18372</td>
<td>Missing camel-kamelets-utils-1.0.0.fuse-800050-redhat-00002.jar</td>
</tr>
<tr>
<td>ENTESB-18510</td>
<td>Camel K runtime tests are failing for version 1.9.0.fuse-800037-redhat-00001</td>
</tr>
<tr>
<td>ENTESB-18511</td>
<td>Some kamelets are missing camel-k-kamelet-reify 1.9.0.fuse-800037-redhat-00001 dependency</td>
</tr>
<tr>
<td>ENTESB-18605</td>
<td>CVE respin: cyrus-sasl security update RHSA :88692 Important Due: 03/25/2022</td>
</tr>
<tr>
<td>ENTESB-18621</td>
<td>Can’t upgrade Camel-K operator from 1.6.3 → 1.6.4</td>
</tr>
<tr>
<td>ENTESB-18624</td>
<td>Salesforce-source kamelet doesnt work in CK3</td>
</tr>
</tbody>
</table>
CHAPTER 5. SERVICE REGISTRY RELEASE NOTES

Red Hat Integration - Service Registry 2.0 is available as a General Availability component in Red Hat Integration 2022.Q1. Service Registry is a datastore for standard event schemas and API designs, which is based on the [Apicurio Registry](https://www.apicurio.org) open source community project.

You can use Service Registry to manage and share the structure of your data using a web console, REST API, Maven plug-in, or Java client. For example, client applications can dynamically push or pull the latest schema updates to or from Service Registry without needing to redeploy. You can also use Service Registry to create optional rules to govern how registry content evolves over time. For example, this includes rules for content validation or for backwards and forwards compatibility of schema or API versions.

5.1. SERVICE REGISTRY INSTALLATION OPTIONS

You can install Service Registry on OpenShift with either of the following data storage options:

- AMQ Streams
- PostgreSQL database

For more details, see [Installing and deploying Service Registry on OpenShift](https://example.com/)

5.2. SERVICE REGISTRY PLATFORM COMPONENT VERSIONS

Service Registry 2.0.3 supports the following versions:

- OpenShift Container Platform 4.9 or 4.6
- OpenJDK 11
- AMQ Streams 1.8
- PostgreSQL 12
- Debezium 1.4
- Camel Kafka Connector - Technology Preview

5.3. SERVICE REGISTRY NEW FEATURES

Service Registry security

- *Authentication based on Red Hat Single Sign-On* - optionally protect the registry so that its REST API requires users to authenticate (OAuth and HTTP basic are supported)
- *Role-based authorization* - when authentication is enabled, users must have at least one role of `sr-admin`, `sr-developer`, or `sr-readonly`
- *Creator-only authorization* - option to prevent changes to artifacts unless the authenticated user originally created the artifact
- *Kafka OAuth authentication* - for storage in AMQ Streams, you can configure access to a Kafka cluster that requires OAuth authentication
Service Registry core

- **Registry artifact groups** - optionally organize schema and API artifacts into custom named logical groupings
- **Refactored Kafka serializer/deserializer (SerDes) classes** - significant updates to the Java SerDes layer to address ease of use, consistency, and functionality
- **Event sourcing** - option to configure the registry to trigger events whenever a change is made, based on the CloudEvents specification

Service Registry data storage

- **SQL-based storage** - new SQL storage implementation with support for PostgreSQL database
- **Kafka-based storage** - new hybrid storage using AMQ Streams to store artifact data and an embedded SQL database to represent it in memory

Service Registry v2 REST API

- **Custom versioning** - option to provide a custom version number when using the REST API to create or update artifacts
- **Improved artifact searching** - updates to the REST API to allow improved searching of artifacts
- **Import/export API** - updates to the REST API with operations to export and import registry data in .zip format
- **CNCF Schema Registry API support** - implementation of the Cloud Native Computing Foundation Schema Registry REST API

**NOTE**

The Service Registry v2 REST API is compatible with the Confluent Schema Registry REST API, which does not include the new artifact groups. Backwards compatibility is also maintained with the existing Service Registry v1 REST API.

Service Registry Operator

- **Improved performance and streamlining** - Operator uses Deployment on OpenShift (instead of DeploymentConfig), predictable resource naming (no random suffixes), and resources created in parallel.
- **Registry data storage** - support for the new SQL and Kafka-based storage options.
- **ApicurioRegistry CRD v1** - uses standardized conditions field in the status block to better indicate issues or errors in the Operator or the application.
- **Multi-namespace deployment** - When the Operator is installed in a namespace, it can watch all namespaces (or a selected subset), so applications can be deployed in any or multiple namespaces.
• **Disconnected installation** – Support for installing on OpenShift in a restricted network was added in versions 2.0.1 and 1.1.2. For more details, see [Mirroring images for a disconnected installation](#).

**Service Registry user documentation and examples**

• Documentation library updated with new features in version 2.0:
  - Installing and deploying Service Registry on OpenShift
  - Migrating Service Registry deployments
  - Service Registry User Guide
  - Registry v2 core REST API documentation

• Updated Open source demonstration applications:
  - [https://github.com/Apicurio/apicurio-registry-examples](https://github.com/Apicurio/apicurio-registry-examples)

**5.4. SERVICE REGISTRY DEPRECATED AND REMOVED FEATURES**

**Service Registry deprecated features**

• Service Registry version 1.x has been deprecated in version 2.0 and will soon go out of full support. For more details, see the [Red Hat Middleware Product Update and Support Policy](#).

**Service Registry removed features**

• Infinispan cache-based storage option has been removed

• Java Persistence API (JPA) storage option has been replaced by the new PostgreSQL database storage option

• Kafka-based storage option in AMQ Streams has been replaced by the new hybrid storage option in AMQ Streams with in-memory H2 database

• Service Registry Java client no longer supports OpenJDK 8 and supports OpenJDK 11 instead

**5.5. MIGRATING SERVICE REGISTRY DEPLOYMENTS**

For details migrating from Service Registry version 1.1 to 2.x, see [Migrating Service Registry deployments](#).

For details on migrating registry data between Service Registry version 2.x instances, see [Exporting and importing registry content using the Registry REST API](#).

**5.6. SERVICE REGISTRY RESOLVED ISSUES**

**Service Registry core resolved issues**

**IPT-651** – Error deserializing Protobuf message when getting schema from Service Registry
The Kafka producer application can set the schema, but the consumer application fails to get the schema from the registry and raises an `org.apache.kafka.common.errors.SerializationException` with an error such as:

```
Error deserializing Protobuf message for id 3
Caused by: java.io.IOException: Invalid schema syntax = "proto3":
|package ...commons;|
|import "head.proto";|
|option java_package = "packaget";|
|option java_multiple_files = true;|
|message AuditMessage {|
|commons.Head|
|head = 1;|
|int64 id = 5;|
|string user = 6;|
|bytes extraData = 7;|
|string signature = 8;|
}|
with refs [] of type AVRO
```

**IPT-625 - Error uploading artifact to Service Registry with KafkaSQL storage**

When Service Registry is installed with the KafkaSQL storage option, an `io.apicurio.registry.storage.RegistryStorageException` is raised when a new artifact is uploaded. Possible error messages include `SQL error: Expected one element, but found none.`

This error was caused by Kafka log compaction, which removed messages that control database sequences. This issue is now fixed by preventing the messages that control the database sequence from being compacted. For more details, see the Apicurio community blog post on Resolving a bug in KafkaSQL storage for Apicurio Registry.

**IPT-159 - Registry v1 API and Confluent compatibility API mismatch**

Existing users migrating to Service Registry v2.x were required to upgrade all of their Kafka client applications that use Service Registry v1 serializers/deserializers (SerDes) to use the Service Registry v2 SerDes instead.

Service Registry provides a new environment variable named `ENABLE_CCOMPAT_LEGACY_ID_MODE` that you can use to revert to the legacy behavior of the v1 compatibility API. When this variable is set to true, Service Registry uses `globalId` instead of `contentId` as the unique integer identifier for schemas uploaded using the compatibility API.

**Registry-1619 - Service Registry server cannot be properly configured to require authentication without role-based authorization**

When role-based authorization is disabled in the Service Registry server, authentication is effectively also disabled. Even when OpenID Connect is enabled in Quarkus, users are not required to provide credentials. If a user provides invalid credentials, a request fails. However, if a user provides no credentials, the request succeeds on behalf of an anonymous user. And because roles are disabled, no additional checking is done.

**Registry-1289 - Registry does not work on IPv6**

When trying to deploy Service Registry using the Operator on a Kubernetes server with Internet Protocol v6, the registry server fails to start.

**Registry-1151 - Error fetching JavaScript libraries when running in a closed network**

When running in a closed network, the Redoc JavaScript libraries do not load properly because they reference a CDN rather than get included or bundled in the application.

**Registry-1007 - Registry REST API returns 406 error**

The Registry REST API returns a 406 error when the `Accept: application/json` header is included in the request.

**Registry-711 - Service Registry client does not work with Jersey HTTP client**
When the Jersey and RESTEasy JAX-RS providers are both in the classpath, RESTEasy takes precedence and breaks other HTTP client functionality relying on Jersey client support for the `application/octet-stream` transport, which RESTEasy does not seem to support.

Service Registry Operator resolved issues

**Operator-41 - Example CRD is empty**

The provided example `ApicurioRegistry` custom resource definition should not be empty.

### 5.7. SERVICE REGISTRY KNOWN ISSUES

#### Service Registry core known issues

**Registry-2394 - Service Registry API endpoint for core v1 compatibility not properly protected by authentication**

The legacy `MY-REGISTRY-URL/api/` endpoint is an alias for the v1 core registry API that moved to `MY-REGISTRY-URL/apis/registry/v1` in Service Registry v2.x. This legacy endpoint is currently not protected when authentication is configured. This issue is fixed in Apicurio Registry v2.1.x, where the authentication layer no longer uses the Quarkus policies configured in `application.properties`.

To work around this issue in Service Registry v2.0.x, disable the core v1 legacy endpoint by setting the `REGISTRY_DISABLE_APIS` environment variable to a value of `/apis/ibmcompat/./api/`.

**IPT-701 - CVE-2022-23221 H2: Loading of custom classes from remote servers through JNDI**

When Service Registry data is stored in AMQ Streams, the H2 database console allows remote attackers to execute arbitrary code using the JDBC URL. Service Registry is not vulnerable by default and a malicious configuration change is required.

#### Service Registry operator known issues

**Operator-42 - Auto-generation of OpenShift route may use wrong base host value**

The auto-generation of the Service Registry OpenShift route may use a wrong base host value if there are multiple `routerCanonicalHostname` values.

**Operator-32 - Operator should support SCRAM authorization without TLS, not only SCRAM+TLS**

The Service Registry Operator should support Salted Challenge Response Authentication Mechanism (SCRAM) authorization without Transport Layer Security (TLS), not only SCRAM+TLS.
CHAPTER 6. RED HAT INTEGRATION OPERATORS

Red Hat Integration 2022.Q1 introduces Red Hat Integration Operator 1.3.

Red Hat Integration provides Operators to automate the deployment of Red Hat Integration components on OpenShift. You can use Red Hat Integration Operator to manage those Operators.

Alternatively, you can manage each component Operator individually. This section introduces Operators and provides links to detailed information on how to use Operators to deploy Red Hat Integration components.

6.1. WHAT OPERATORS ARE

Operators are a method of packaging, deploying, and managing a Kubernetes application. They take human operational knowledge and encode it into software that is more easily shared with consumers to automate common or complex tasks.

In OpenShift Container Platform 4.x, the Operator Lifecycle Manager (OLM) helps users install, update, and generally manage the life cycle of all Operators and their associated services running across their clusters. It is part of the Operator Framework, an open source toolkit designed to manage Kubernetes native applications (Operators) in an effective, automated, and scalable way.

The OLM runs by default in OpenShift Container Platform 4.x, which aids cluster administrators in installing, upgrading, and granting access to Operators running on their cluster. The OpenShift Container Platform web console provides management screens for cluster administrators to install Operators, as well as grant specific projects access to use the catalog of Operators available on the cluster.

OperatorHub is the graphical interface that OpenShift cluster administrators use to discover, install, and upgrade Operators. With one click, these Operators can be pulled from OperatorHub, installed on the cluster, and managed by the OLM, ready for engineering teams to self-service manage the software in development, test, and production environments.

Additional resources

- For more information about Operators, see the OpenShift documentation.

6.2. RED HAT INTEGRATION COMPONENT OPERATORS

You can install and upgrade each Red Hat Integration component Operator individually, for example, using the 3scale Operator, the Camel K Operator, and so on.

6.2.1. 3scale Operators

- 3scale Operator
- 3scale APIcast Operator

6.2.2. AMQ Operators

- AMQ Broker Operator
- AMQ Interconnect Operator
• AMQ Streams Cluster Operator
• AMQ Online Operator

6.2.3. Camel K Operator
• Camel K Operator - Technology Preview

6.2.4. Fuse Operators
• Fuse on OpenShift - Samples Operator
• Fuse on OpenShift - Fuse Console Operator
• Fuse on OpenShift - API Designer Operator
• Fuse Online Operator

6.2.5. Service Registry Operator
• Service Registry Operator

6.3. RED HAT INTEGRATION OPERATOR (DEPRECATED)
You can use Red Hat Integration Operator 1.3 to install and upgrade multiple Red Hat Integration component Operators:
• 3scale
• 3scale APIcast
• AMQ Broker
• AMQ Interconnect
• AMQ Streams
• API Designer
• Camel K
• Fuse Console
• Fuse Online
• Service Registry

NOTE
The Red Hat Integration Operator has been deprecated and will be removed in the future. It will be available from the OperatorHub in OpenShift 4.6 to 4.10. The individual Red Hat Integration component Operators will continue to be supported, which you can install separately.

6.3.1. Supported components
Before installing the Operators using Red Hat Integration Operator 1.3, check the updates in the Release Notes of the components. The Release Notes for the supported version describe any additional upgrade requirements.

- Release Notes for Red Hat 3scale API Management 2.10 On-premises
- Release Notes for Red Hat AMQ Broker 7.8
- Release Notes for Red Hat AMQ Interconnect 1.10
- Release Notes for Red Hat AMQ Streams 2.0 on OpenShift
- Release Notes for Red Hat Fuse 7.10 (Fuse and API Designer)
- Release Notes for Red Hat Integration 2021.Q3 (Red Hat Integration - Service Registry 2.0 release notes)
- Release Notes for Red Hat Integration 2021.Q4 (Camel K release notes)

AMQ Streams new API version

Red Hat Integration Operator 1.3 installs the Operator for AMQ Streams 2.0.

You must upgrade your custom resources to use API version \texttt{v1beta2} before upgrading to AMQ Streams version 1.8 or later.

AMQ Streams 1.7 introduced the \texttt{v1beta2} API version, which updates the schemas of the AMQ Streams custom resources. Older API versions are now deprecated. After you have upgraded to AMQ Streams 1.7, and before you upgrade to AMQ Streams 2.0, you must upgrade your custom resources to use API version \texttt{v1beta2}.

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 \texttt{v1beta2}
3. Upgrade to AMQ Streams 2.0

For more information, refer to the following documentation:

- Upgrade requirements
- Introducing the \texttt{v1beta2} API version.
Upgrade of the AMQ Streams Operator to version 2.0 will fail in clusters if custom resources and CRDs haven’t been converted to version v1beta2. The upgrade will be stuck on Pending. If this happens, do the following:

1. Perform the steps described in the Red Hat Solution, Forever pending cluster operator upgrade.

2. Scale the Integration Operator to zero, and then back to one, to trigger an installation of the AMQ Streams 2.0 Operator.

Service Registry 2.0 migration

Red Hat Integration Operator installs Red Hat Integration - Service Registry 2.0.

Service Registry 2.0 does not replace Service Registry 1.x installations, which need to be manually uninstalled.

For information on migrating from Service Registry version 1.x to 2.0, see the Service Registry 2.0 release notes.

6.3.2. Support life cycle

To remain in a supported configuration, you must deploy the latest Red Hat Integration Operator version. Each Red Hat Integration Operator release version is only supported for 3 months.

6.3.3. Fixed issues

There are no fixed issues for Red Hat Integration Operator 1.3.

Additional resources

- For more details on managing multiple Red Hat Integration component Operators, see Installing the Red Hat Integration Operator on OpenShift.