Red Hat Integration 2022.Q2

Release Notes for Red Hat Integration 2022.Q2

What's new in Red Hat Integration

Last Updated: 2022-06-16

Red Hat Integration 2022.Q2 Release Notes for Red Hat Integration 2022.Q2

What's new in Red Hat Integration

Legal Notice

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

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:

```
<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:

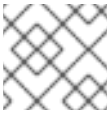
- Apache Camel 3.11.2 Release Notes
- Apache Camel 3.11.3 Release Notes
- Apache Camel 3.11.4 Release Notes
- Apache Camel 3.11.5 Release Notes

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

The quarkus-bom still manages **org.apache.logging.log4j:log4j-api**.

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:

```
{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:

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

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

When running a project in Quarqus 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 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

- Getting Started with Debezium
- Debezium User Guide

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:

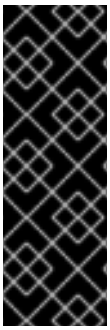
New deployment mechanism

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. 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
- Support for IBM MQ connector
- Support for Oracle 19 database

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

- Operator bundle format in the OpenShift documentation .

4.3. IMPORTANT NOTES

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

Support for IBM MQ source connector in Camel K

IBM MQ source connector kamelet is added to latest Camel K.

Support for Oracle 19

Oracle 19 is now supported in Camel K. Refer Supported configurations page for more information.

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).

Name	Package
AWS 2 Kinesis	**camel-quarkus-aws2-kinesis**
AWS 2 Lambda	**camel-quarkus-aws2-lambda**
AWS 2 S3 Storage Service	**camel-quarkus-aws2-s3**
AWS 2 Simple Notification System (SNS)	**camel-quarkus-aws2-sns**
AWS 2 Simple Queue Service (SQS)	**camel-quarkus-aws2-sqs**
File	**camel-quarkus-file**
FTP	**camel-quarkus-ftp**

Name	Package
FTPS	**camel-quarkus-ftp**
SFTP	**camel-quarkus-ftp**
HTTP	**camel-quarkus-http**
JMS	**camel-quarkus-jms**
Kafka	**camel-quarkus-kafka**
Kamelets	**camel-quarkus-kamelet**
Metrics	**camel-quarkus-microprofile-metrics**
MongoDB	**camel-quarkus-mongodb**
Salesforce	**camel-quarkus-salesforce**
SQL	**camel-quarkus-sql**
Timer	**camel-quarkus-timer**

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).

Name	Package
Avro	**camel-quarkus-avro**
Bindy (for CSV)	**camel-qaurkus-bindy**
JSON Jackson	**camel-quarkus-jackson**
Jackson Avro	**camel-quarkus-jackson-avro**

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:

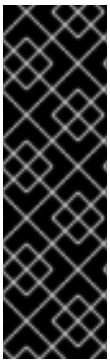
- Builder trait
- Camel trait
- Container trait
- Dependencies trait
- Deployer trait
- Deployment trait
- Environment trait
- Jvm trait
- Kamelets trait
- Owner trait
- Platform trait
- Pull Secret trait
- Prometheus trait
- Quarkus trait
- Route trait
- Service trait
- Error Handler trait

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/2022.q1/html-single/integrating_applications_with_kamelets.



IMPORTANT

Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them 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

Kamelet	File name	Type (Sink, Source, Action)
Avro Deserialize action	**avro-deserialize-action.kamelet.yaml**	Action (data conversion)
Avro Serialize action	**avro-serialize-action.kamelet.yaml**	Action (data conversion)
AWS Redshift sink	**aws-redshift-sink.kamelet.yaml**	Sink
AWS 2 Kinesis sink	**aws-kinesis-sink.kamelet.yaml**	Sink
AWS 2 Kinesis source	**aws-kinesis-source.kamelet.yaml**	Source
AWS 2 Lambda sink	**aws-lambda-sink.kamelet.yaml**	Sink
AWS 2 Simple Notification System sink	**aws-sns-sink.kamelet.yaml**	Sink
AWS 2 Simple Queue Service sink	**aws-sqs-sink.kamelet.yaml**	Sink
AWS 2 Simple Queue Service source	**aws-sqs-source.kamelet.yaml**	Source
AWS 2 Simple Queue Service FIFO sink	**aws-sqs-fifo-sink.kamelet.yaml**	Sink
AWS 2 S3 sink	**aws-s3-sink.kamelet.yaml**	Sink
AWS 2 S3 source	**aws-s3-source.kamelet.yaml**	Source

Kamelet	File name	Type (Sink, Source, Action)
AWS 2 S3 Streaming Upload sink	**aws-s3-streaming-upload-sink.kamelet.yaml**	Sink
Cassandra sink (Technology Preview)	**cassandra-sink.kamelet.yaml**	Sink
Cassandra source (Technology Preview)	**cassandra-source.kamelet.yaml**	Source
Elasticsearch Index sink (Technology Preview)	**elasticsearch-index-sink.kamelet.yaml**	Sink
Extract Field action	**extract-field-action.kamelet.yaml**	Action
FTP sink	**ftp-sink.kamelet.yaml**	Sink
FTP source	**ftp-source.kamelet.yaml**	Source
Has Header Key Filter action	**has-header-filter-action.kamelet.yaml**	Action (data transformation)
Hoist Field action	**hoist-field-action.kamelet.yaml**	Action
HTTP sink	**http-sink.kamelet.yaml**	Sink
Insert Field action	**insert-field-action.kamelet.yaml**	Action (data transformation)
Insert Header action	**insert-header-action.kamelet.yaml**	Action (data transformation)
Is Tombstone Filter action	**is-tombstone-filter-action.kamelet.yaml**	Action (data transformation)
Jira source (Technology Preview)	**jira-source.kamelet.yaml**	Source
JMS sink	**jms-amqp-10-sink.kamelet.yaml**	Sink
JMS source	**jms-amqp-10-source.kamelet.yaml**	Source
JMS IBM MQ sink	**jms-ibm-mq-sink.kamelet.yaml**	Sink

Kamelet	File name	Type (Sink, Source, Action)
JMS IBM MQ source	**jms-ibm-mq-source.kamelet.yaml**	Source
JSON Deserialize action	**json-deserialize-action.kamelet.yaml**	Action (data conversion)
JSON Serialize action	**json-serialize-action.kamelet.yaml**	Action (data conversion)
Kafka sink	**kafka-sink.kamelet.yaml**	Sink
Kafka source	**kafka-source.kamelet.yaml**	Source
Kafka Topic Name Filter action	**topic-name-matches-filter-action.kamelet.yaml**	Action (data transformation)
Log sink (for development and testing purposes)	**log-sink.kamelet.yaml**	Sink
MariaDB sink	**mariadb-sink.kamelet.yaml**	Sink
Mask Fields action	**mask-field-action.kamelet.yaml**	Action (data transformation)
Message TimeStamp Router action	**message-timestamp-router-action.kamelet.yaml**	Action (router)
MongoDB sink	**mongodb-sink.kamelet.yaml**	Sink
MongoDB source	**mongodb-source.kamelet.yaml**	Source
MySQL sink	**mysql-sink.kamelet.yaml**	Sink
PostgreSQL sink	**postgresql-sink.kamelet.yaml**	Sink
Predicate filter action	**predicate-filter-action.kamelet.yaml**	Action (router/filter)
Protobuf Deserialize action	**protobuf-deserialize-action.kamelet.yaml**	Action (data conversion)
Protobuf Serialize action	**protobuf-serialize-action.kamelet.yaml**	Action (data conversion)

Kamelet	File name	Type (Sink, Source, Action)
Regex Router action	**regex-router-action.kamelet.yaml**	Action (router)
Replace Field action	**replace-field-action.kamelet.yaml**	Action
Salesforce source	**salesforce-source.kamelet.yaml**	Source
SFTP sink	**sftp-sink.kamelet.yaml**	Sink
SFTP source	**sftp-source.kamelet.yaml**	Source
Slack source	**slack-source.kamelet.yaml**	Source
SQL Server Database sink	**sqlserver-sink.kamelet.yaml**	Sink
Telegram source (Technology Preview)	**telegram-source.kamelet.yaml**	Source
Timer source (for development and testing purposes)	**timer-source.kamelet.yaml**	Source
TimeStamp Router action	**timestamp-router-action.kamelet.yaml**	Action (router)
Value to Key action	**value-to-key-action.kamelet.yaml**	Action (data transformation)

4.6. CAMEL K KNOWN ISSUES

The following known issues apply to the Camel K:

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:

```
$ 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 .

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 work around is to add the atlassian repo manually.

```
apiVersion: camel.apache.org/v1
kind: 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 work around 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 1.6.6.

- Section 4.7.1, "Enhancements in Camel K 1.6.6"
- Section 4.7.2, "Bugs resolved in Camel K 1.6.6"

4.7.1. Enhancements in Camel K 1.6.6

The following table lists the enhancements in Camel K 1.6.6.

Table 4.2. Camel K 1.6.6 Enhancements

Issue	Description
ENTESB-16551	Revisit KNative documentation from the Camel-K pov
ENTESB-17296	Devise a testing strategy for Quickstarts and Kamelets
ENTESB-18352	Add AWS Redshift sink Kamelet
ENTESB-19069	Camel K 1.6.x to bring in the new base layer images for Openjdk and Zlib

4.7.2. Bugs resolved in Camel K 1.6.6

The following table lists the resolved bugs in Camel K 1.6.6.

Table 4.3. Camel K 1.6.6 Resolved Bugs

Issue	Description
ENTESB-15948	camel-slack producer fails after 1st message sent
ENTESB-17058	eventstreaming quickstart: yaks test fails
ENTESB-17114	kamel get reports running status when the pod is not running
ENTESB-17338	kameletBinding GoogleSheetsSource delay parameter malfunctioning
ENTESB-17967	Kamelets: Elasticsearch index sink indexId random generator for null
ENTESB-18304	Cassandra sink kamelet fails with unknown property
ENTESB-18305	Default value for Consistency in Cassandra source is not usable
ENTESB-18542	wrong image with kamel install --olm=false
ENTESB-18554	Quickstart camel-k-example-event-streaming creates invalid kafka instance
ENTESB-18559	Typo in yaks test OpenAQConsumer.feature of quickstart Camel K: Event Streaming Example
ENTESB-18654	wrong XSD url generated with XML initialiser
ENTESB-18951	Camel-k operator image grade B

CHAPTER 5. RED HAT INTEGRATION OPERATORS

Red Hat Integration 2022.Q2 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.

5.1. 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.

5.2. 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.

5.2.1. 3scale Operators

- 3scale Operator
- 3scale APIcast Operator

5.2.2. AMQ Operators

- AMQ Broker Operator
- AMQ Interconnect Operator

- AMQ Streams Cluster Operator
- AMQ Online Operator

5.2.3. Camel K Operator

- Camel K Operator - Technology Preview

5.2.4. Fuse Operators

- Fuse on OpenShift - Samples Operator
- Fuse on OpenShift - Fuse Console Operator
- Fuse on OpenShift - API Designer Operator
- Fuse Online Operator

5.2.5. Service Registry Operator

- Service Registry Operator

5.3. 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.

5.3.1. 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 **v1beta2** before upgrading to AMQ Streams version 1.8 or later.

AMQ Streams 1.7 introduced the **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 **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 v1beta2
3. Upgrade to AMQ Streams 2.0

For more information, refer to the following documentation:

- Upgrade requirements
- Introducing the v1beta2 API version.

WARNING

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.

5.3.2. 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.

5.3.3. 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.