Red Hat Integration 2021.Q3

Release Notes for Red Hat Integration 2021.Q3

What's new in Red Hat Integration

Last Updated: 2021-12-10

Red Hat Integration 2021.Q3 Release Notes for Red Hat Integration 2021.Q3

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. NEW FEATURES IN THIS RELEASE

This section provides a summary of the key new features in Red Hat Integration 2021.Q3 and provides links to more details on new features available in different components.

NOTE

These release notes include details on components updated in Red Hat Integration 2021.Q3 only. For details on the latest versions of other components, such as Debezium, Camel K, and Camel Quarkus, see the Red Hat Integration Release Notes for 2021-Q1 and Red Hat Integration Release Notes for 2021-Q2 .

2.1. NEW INTEGRATION FEATURES

Serverless Camel K

- Cloud-native integration for serverless architectures based on Camel 3.10 and Kamelets in Red Hat Integration - Camel K 1.4 General Availability

Camel Quarkus extensions

- Camel components available as Camel Quarkus extensions in Camel Quarkus Technology Preview.

Data integration

- Change data capture and real-time events, including new DB2 connector and integration with Service Registry in Debezium 1.5

Kafka schema registry

- Improved Apache Kafka schema registry and API registry with OpenShift Operator in Service Registry 2.0 General Availability

Red Hat Integration Operator

- Install and upgrade the OpenShift Operators that manage your Red Hat Integration components with Red Hat Integration Operator 1.1

2.2. NEW COMPONENT FEATURES

For more details on what's new in Red Hat Integration 2021.Q3 components:

- Red Hat 3scale API Management
  - Red Hat 3scale API Management 2.10 On-Premises Release Notes
  - Red Hat 3scale API Management SaaS Release Notes
- Red Hat AMQ Product Documentation
- Red Hat Fuse 7.9 Release Notes

CHAPTER 3. CAMEL QUARKUS RELEASE NOTES

Camel Quarkus is available as a Technology Preview component in Red Hat Integration 2021.Q3.

NOTE

In this Technology Preview, Camel Quarkus is supported in JVM mode only.

Camel Quarkus brings the integration capabilities of Apache Camel and its vast component library to the Quarkus runtime. By using Camel Quarkus, you can to take advantage of the performance benefits, developer joy and the container first ethos which Quarkus provides.

Camel Quarkus provides Quarkus extensions which are units of the Quarkus distribution. Extensions configure, boot and integrate Camel components in your Quarkus application and are typically configured as dependencies in your project.

Camel Quarkus takes advantage of the many performance improvements made in Camel 3, which results in a lower memory footprint, less reliance on reflection and faster startup times.

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.

This Technology Preview feature provides early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. For more information about support scope, see Technology Preview Features Support Scope.

3.1. CAMEL QUARKUS FEATURES

The Camel Quarkus Technology Preview provides the following main features:

3.1.1. Platform and core component versions

- OpenShift Container Platform 4.6 or 4.7
- Red Hat build of Quarkus 1.11.7
- Apache Camel 3.10
- Apache Camel Quarkus 1.8.1
- OpenJDK 11

3.1.2. Technology Preview 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.

Highly configurable

All of the important aspects of a Camel Quarkus application can be set up programatically 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

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

3.1.3. Available extensions in Technology Preview

- camel-quarkus-aws2-ddb
- camel-quarkus-aws2-kinesis
- camel-quarkus-aws2-lambda
- camel-quarkus-aws2-s3
- camel-quarkus-aws2-sns
- camel-quarkus-aws2-sqs
- camel-quarkus-bean
- camel-quarkus-core
- camel-quarkus-direct
- camel-quarkus-elasticsearch-rest
- camel-quarkus-file
- camel-quarkus-ftp
- camel-quarkus-http
- camel-quarkus-jira
- camel-quarkus-jms
- camel-quarkus-jta
- camel-quarkus-kafka
- camel-quarkus-kamelet
- camel-quarkus-log
- camel-quarkus-main
- camel-quarkus-microprofile-health
- camel-quarkus-microprofile-metrics

- camel-quarkus-mllp
- camel-quarkus-mock
- camel-quarkus-mongodb
- camel-quarkus-netty
- camel-quarkus-platform-http
- camel-quarkus-rest
- camel-quarkus-salesforce
- camel-quarkus-saxon
- camel-quarkus-seda
- camel-quarkus-sql
- camel-quarkus-timer
- camel-quarkus-xpath

3.1.4. Available data formats in Technology Preview

- camel-quarkus-avro
- camel-quarkus-jackson-avro
- camel-quarkus-bindy
- camel-quarkus-hl7
- camel-quarkus-jackson
- camel-quarkus-jacksonxml
- camel-quarkus-jackson-protobuf
- camel-quarkus-soap

3.1.5. Available lanuages in Technology Preview

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

- JSON Path

Additional resources

- Camel Extensions for Quarkus
- Developing applications with Camel Quarkus
- Getting Started with Camel Quarkus Extensions

3.1.6. Important notes

Important notes for the Integration TP2 release of the Camel Quarkus extensions distribution:

Red Hat Enterprise Linux (RHEL) support

RHEL 8 is supported in this TP2 release.

SOAP proxy example not supported

The proxy example referred to in the upstream link from the Camel Quarkus SOAP data format documentation is not supported in this TP2 release.

CHAPTER 4. 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 4.1, "Debezium database connectors"
- Section 4.2, "Debezium supported configurations"
- Section 4.3, "Debezium installation options"
- Section 4.4, "New Debezium features"

4.1. DEBEZIUM DATABASE CONNECTORS

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

- Db2
- MongoDB
- MySQL
- Oracle (Developer 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

4.2. DEBEZIUM SUPPORTED CONFIGURATIONS

For information about Debezium supported configurations, including information about supported database versions, see the Debezium 1.5 Supported configurations page .

AMQ Streams new API version

Debezium runs on AMQ Streams 1.8.

AMQ Streams now supports 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 1.8, you must upgrade your custom resources to use API version **v1beta2**.

For more information, see the Debezium User Guide.

4.3. DEBEZIUM INSTALLATION OPTIONS

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

- Installing Debezium on OpenShift
- Installing Debezium on RHEL

IMPORTANT

Technology Preview and Developer 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 or Developer Preview features in production environments. Technology Preview and Developer Preview feature 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 .

4.4. NEW DEBEZIUM FEATURES

Debezium 1.5 includes the following updates:

Promoted to GA

The following features that were offered as Technology Previews in the previous release are now available for General Availability:

Installing Debezium on RHEL

Debezium on Red Hat Enterprise Linux.

Technology Preview features

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.

Developer Preview features

Debezium Oracle connector

Connector for Oracle Database. In this release the connector provides the following capabilities:

- 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).

Debezium documentation

- Information about how to use predicates to apply SMTs selectively to messages: Applying transformations selectively with SMT predicates.

CHAPTER 5. 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.

5.1. 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
- The following traits are provided as Technology Preview:
  - Knative Service trait
  - Knative trait
  - OpenAPI trait

5.1.1. 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-1.4

For information about how to use kamelets to connect applications and services, see https://access.redhat.com/documentation/en-us/red_hat_integration/2021.q3/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 5.1. 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 2 S3 sink	**aws-s3-sink.kamelet.yaml**	Sink
AWS 2 S3 source	**aws-s3-source.kamelet.yaml**	Source
AWS 2 S3 Streaming Upload sink	**aws-s3-streaming-upload-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 SQS FIFO sink	**aws-sqs-fifo-sink.kamelet.yaml**	Sink
Cassandra sink*	**cassandra-sink.kamelet.yaml**	Sink
Cassandra source*	**cassandra-source.kamelet.yaml**	Source

Kamelet	File name	Type (Sink, Source, Action)
Elasticsearch Index sink*	**elasticsearch-index-sink.kamelet.yaml**	Sink
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)
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*	**jira-source.kamelet.yaml**	Source
JMS sink	**jms-amqp-10-sink.kamelet.yaml**	Sink
JMS source	**jms-amqp-10-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)

Kamelet	File name	Type (Sink, Source, Action)
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)
Regex Router action	**regex-router-action.kamelet.yaml**	Action (router)
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*	**telegram-source.kamelet.yaml**	Source
TimeStamp Router action	**timestamp-router-action.kamelet.yaml**	Action (router)

Kamelet	File name	Type (Sink, Source, Action)
Value to Key action	**value-to-key-action.kamelet.yaml**	Action (data transformation)

5.2. 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

5.2.1. 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 .

5.3. IMPORTANT NOTES

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

Supported Enterprise Integraion 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 1.4, but the error messaging for incorrect YAML DSL code is still in development.

JAVA DSL Limitations

Java DSL in Camel K 1.4 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 1.4.

5.3.1. 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).

5.3.1.1. 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**
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**

Name	Package
SQL	**camel-quarkus-sql**
Timer	**camel-quarkus-timer**

5.3.1.2. 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**

5.3.1.3. 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

5.3.1.4. 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

5.4. CAMEL K KNOWN ISSUES

The following known issues apply to the Camel K 1.4:

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-15893 - Camel K catalog contains camel-quarkus-spark reference and cannot deploy integrations with Apache Spark

The Camel K catalog includes the **camel-quarkus-spark** component, which is no longer included the in the Bill of Materials (BOM) for Camel Quarkus extensions. When you try to deploy a Camel K integration using the Spark component in Camel Quarkus, the integration cannot be compiled due to this missing dependency.

For more details, see the Spark component in Camel Quarkus .

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-17071 - Azure storage kamelets doesn’t work

Azure storage kamelet produces an error when binding. The workaround is to copy the azure storage kamelet from Kamelet catalog and replace it with the kamelet that oprator installs by default. Then follow the instructions in the Kamelets user guide.

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.

2492 - Within a kamelet binding, Camel components in URIs require dependency declarations

The Camel K operator generates an integration from a kamelet binding at runtime. For integrations not generated by a kamelet binding, Camel K automatically handles the dependency management and imports all the required libraries from the Camel catalog. However, you must specify any Camel components that you reference in a URI within a kamelet binding as a dependency.

5.5. CAMEL K FIXED ISSUES

The following sections list the issues that have been fixed in Camel K 1.4.0.

- Section 5.5.1, “Enhancements in Camel K 1.4.0”
- Section 5.5.2, “Feature requests in Camel K 1.4.0”
- Section 5.5.3, “Bugs resolved in Camel K 1.4.0”

5.5.1. Enhancements in Camel K 1.4.0

The following table lists the enhancements in Camel K 1.4.0.

Table 5.2. Camel K 1.4.0 Enhancements

Issue	Description
ENTESB-14370	Seamless upgrades for Camel-K
ENTESB-16956	"Missing tests for ""pull secret"" trait properties"
ENTESB-16567	Document how to debug Kamelets
ENTESB-16351	Cleanup configuration options for Camel K
ENTESB-16316	Backport CAMEL-16474 : camel-azure - Blob consumer should extend scheduled endpoint
ENTESB-14412	Possibility to specify operator requests and limits
ENTESB-14218	[Observability] Provide ootb metrics for Camel K
ENTESB-16348	Support for Route Template local beans

5.5.2. Feature requests in Camel K 1.4.0

The following table lists the features requests in Camel K 1.4.0.

Table 5.3. Camel K 1.4.0 Feature Requests

Issue	Description
ENTESB-16704	Create MongoDB kamelet
ENTESB-16701	Kamelets support for custom beans and drivers
ENTESB-16420	Http sink kamelet into downstream catalogue
ENTESB-16318	Backport Streaming upload for Camel AWS2-S3
ENTESB-16435	Provide an example that shows how Camel K integrations can access a JDBC database
ENTESB-16436	Provide an example that shows how Camel K integrations can access a JMS broker
ENTESB-16419	Error handling in Kamelets
ENTESB-16306	Create a Kamelet Catalog

5.5.3. Bugs resolved in Camel K 1.4.0

The following table lists the resolved bugs in Camel K 1.4.0.

Table 5.4. Camel K 1.4.0 Resolved Bugs

Issue	Description
ENTESB-15787	kamel run does not work for remote files on windows
ENTESB-15741	Camel 3 failing tests: infinispan
ENTESB-17032	OLM manual upgrade **techpreview → 1.4.x** does not work
ENTESB-16749	"Unrecognized field ""type"" in KameletSpec in YAKS"
ENTESB-16991	Change the JMS dependencies to inherit clients from org.amqphub.quarkus:quarkus-qpid-jms
ENTESB-16772	Lambda: Using CamelAwsLambdaZipFile header fails at runtime
ENTESB-15278	Camel 3 failing tests: mina
ENTESB-15800	Camel-K: Can't mount secrets with binary data
ENTESB-17113	"Kamel global operator
ENTESB-17076	AMQP Kamelet - unable to find qpid-jms-client dependency
ENTESB-16950	Kameletbinding with aws-kinesis-sink throws NPE if not used insert-header-action kamelet
ENTESB-16930	Port in SQL Kamelet overriden when used with InMemoryChannel source
ENTESB-17005	Catalog.version doesn't have redhat suffix in build 7
ENTESB-17086	Can't upgrade from 1.3.3 to 1.4.0
ENTESB-16882	Failed to run Install Kafka in Kamelet-catalog
ENTESB-16878	Saleforce kamelet return UNAUTHORIZED instead of CREATED
ENTESB-16820	Camel-k-example-knative: Exceptions in cautious-investor pods
ENTESB-16291	Azure Storage Queue: Camel startup fails due to null client in 3.9.0
ENTESB-16317	Backport CAMEL-16498 - Camel-Azure-Storage-queue consumer doesn't work

Issue	Description
ENTESB-15930	Dependency autoloading is not working correctly with YAML format
ENTESB-16068	Kamelets are looked up in github/apache/camel-kamelets in global installation mode with OLM=true
ENTESB-16437	Kamel global operator error executing post actions: error during create resource: Deployment
ENTESB-16430	Pull secret trait does not work as expected
ENTESB-16584	CronJobs written in YAML format don't work correctly
ENTESB-16422	Camel-k-example-saas: loginUrl must be specified with 1.4.0
ENTESB-16432	AWS Kinesis kamelet throws org.apache.camel.NoTypeConversionAvailableException while logging Record
ENTESB-16424	Camel-k-example-api: The specified bucket does not exist with 1.4.0
ENTESB-16289	Camel-K: Integration not marked as Failed when Camel is unable to start
ENTESB-16879	AWS-Kinesis kamelet NoClassDefFoundError: com/fasterxml/jackson/annotation/JsonKey
ENTESB-16887	camel-k-example-event-streaming: sjms2 Error - Unknown parameters=[{ttl=...}]
ENTESB-16900	"prometheus trait: the ""enabled"" property automatically inferred causes error if set by the user"
ENTESB-16999	Remove artemis sink and source kamelets
ENTESB-16998	KameletBinding from KafkaTopic to elasticsearch-index-sink seems to require user parameter
ENTESB-17120	camel-k-example-event-streaming open-aq-consumer is failing with 503 on api.openaq.org
ENTESB-17031	CamelHttpUri in http-sink kamelet overriden if used with InMemoryChannel
ENTESB-16982	**kamel bind** errorHandler is empty
ENTESB-16918	kamel run does not work for remote files on windows (cannot read sources)
ENTESB-17040	Kamelet elasticsearch-index-sink cant use exchangeid as indexid

5.5.4. Bugs resolved in Camel K 1.4.1

The following table lists the resolved bugs in Camel K 1.4.1.

Table 5.5. Camel K 1.4.1 Resolved Bugs

Issue	Description
ENTESB-17309	Camel-K uses a Knative API that has been removed in Serverless 1.17/Knative 0.23

CHAPTER 6. SERVICE REGISTRY RELEASE NOTES

Red Hat Integration - Service Registry 2.0 is available as a General Availability component in Red Hat Integration 2021.Q3. Service Registry is a datastore for standard event schemas and API designs, which is based on the Apicurio Registry 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 backwards and forwards compatibility of schema or API versions.

6.1. 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 .

6.2. SERVICE REGISTRY PLATFORM COMPONENT VERSIONS

Service Registry 2.0.2 supports the following versions:

- OpenShift Container Platform 4.6 or 4.8
- OpenJDK 11
- AMQ Streams 1.8
- PostgreSQL 12
- Debezium 1.4
- Camel Kafka Connector - Technology Preview

6.3. 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

Service Registry core

- *Registry artifact groups* - optionally organize schema and API artifacts into custom named logical groupings
- *Refactored Kafka serializer/deserializer (SerDe) classes* - significant updates to the Java SerDe 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.
- *Registry security* - support for authentication and authorization configuration using Red Hat Single Sign-On.
- *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

- New 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 example CRDs:
  - Red Hat Integration Software Downloads
- New open source demonstration applications:
  - https://github.com/Apicurio/apicurio-registry-examples

6.4. 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

6.5. 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.

6.6. SERVICE REGISTRY RESOLVED ISSUES

Service Registry core resolved issues

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 v2.0.2 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-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 should not be empty

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

6.7. SERVICE REGISTRY KNOWN ISSUES

Service Registry core known issues

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 is caused by Kafka log compaction, which removes messages that control database sequences. The workaround is to disable log compaction and to use the Service Registry export/import feature to force a reset of the database sequences so that the error no longer occurs.

For a detailed explanation, see the Apicurio community blog post on Resolving a bug in KafkaSQL storage for Apicurio Registry.

Registry-1610 - Service Registry web console does not properly disable user roles for authorization

You can configure the Service Registry server to disable role-based authorization. When configured, authentication credentials are required, but roles are ignored, and any authenticated user can preform any action. The web console does not support this, and assumes that if authentication is enabled, role-based authorization is also enabled.

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.

Service Registry operator known issues

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.

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.

CHAPTER 7. RED HAT INTEGRATION OPERATORS

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

7.1. 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.

7.2. RED HAT INTEGRATION OPERATOR

You can use Red Hat Integration Operator 1.2 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

7.2.1. Supported components

Before installing the Operators using Red Hat Integration Operator 1.2, 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 1.8 on OpenShift
- Release Notes for Red Hat Fuse 7.9 (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.Q3 (Camel K release notes)

AMQ Streams new API version

Red Hat Integration Operator 1.1 installed the Operator for AMQ Streams 1.7. Red Hat Integration Operator 1.2 installs the Operator for AMQ Streams 1.8.

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

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 1.8, 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 1.8

For more information, refer to the following documentation:

- Upgrading to AMQ Streams 1.8.0 (Red Hat Solution)
- Upgrade requirements
- Introducing the v1beta2 API version.

WARNING

Upgrade of the AMQ Streams Operator to version 1.8 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 Operator 1.8

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.

7.2.2. Upgrading the AMQ Broker Operator

AMQ Broker now uses a new operator called *AMQ Broker for RHEL 8 (Multiarch)* . Red Hat Integration Operator cannot make an automatic upgrade to the new Operator from the old one.

Instead, you need to perform the following steps to switch to the new Operator.

1. Log in to the OpenShift Container Platform web console as a cluster administrator.
2. Temporarily disable the AMQ Broker Operator in the **Installation** custom resource.
    a. In the left navigation menu, click **Operators** → **Installed Operators**.
    b. Click the **Red Hat Integration Operator** → **Installation** tab → **rhi-installation** → **YAML** tab.
    c. Locate the **amq-broker-installation** field and set the **enabled** property to **false**.
3. Delete the main custom resource instance for the broker deployment in your project. This action deletes the broker deployment.
    a. In the left navigation menu, click **Administration** → **Custom Resource Definitions**
    b. On the **Custom Resource Definitions** page, click the **ActiveMQArtemis** CRD.
    c. Click the **Instances** tab.
    d. Locate the custom resource instance that corresponds to your project namespace.
    e. For your custom resource instance, click the **More Options** icon (three vertical dots) on the right-hand side.

   f. Select **Delete ActiveMQArtemis**.

4. Uninstall the existing AMQ Broker Operator from your project.
   a. In the left navigation menu, click **Operators → Installed Operators**.
   b. From the **Project** drop-down menu at the top of the page, select the project in which you want to uninstall the Operator.
   c. Locate the Red Hat Integration - AMQ Broker instance that you want to uninstall.
   d. For your Operator instance, click the **More Options** icon (three vertical dots) on the right-hand side.
   e. Select **Uninstall Operator**.
   f. On the confirmation dialog box, click **Uninstall**.
5. Re-enable the AMQ Broker Operator in the **Installation** custom resource. Optionally, also change the mode to 'cluster'. This action will trigger the installation of the *AMQ Broker for RHEL 8 (Multiarch)* operator.
6. To recreate your previous broker deployment, create a new custom resource YAML file to match the purpose of your original custom resource and apply it.
   You can base the new custom resource configuration on the sample **broker_activemqartemis_cr.yaml** file. For more information, refer to Deploying a basic broker instance.

7.2.3. 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.

7.2.4. Fixed issues

The following table lists the fixed issues that were affecting Red Hat Integration Operator 1.2. The issues related to OpenShift Container Platform 4.7 and were fixed in the OpenShift Container Platform 4.7.24 bug fix update.

Issue Number	Description
ENTMQST-3047	AMQ Streams Operator 1.7.2 is blocking the installation of other operators.
BZ-1981832	OLM fails with 'ResolutionFailed' found multiple channel heads.
BZ-1989712	OLM fails with 'ResolutionFailed' found multiple channel heads due to deprecated inner channel entry.

Additional resources

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

7.3. 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.

7.3.1. 3scale Operators

- 3scale Operator
- 3scale APIcast Operator

7.3.2. AMQ Operators

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

7.3.3. Camel K Operator

- Camel K Operator - Technology Preview

7.3.4. Fuse Operators

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

7.3.5. Service Registry Operator

- Service Registry Operator

Additional resources

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