Red Hat Integration 2021.Q4

Release Notes for Red Hat Integration 2021.Q4

What's new in Red Hat Integration

Last Updated: 2022-04-05

Red Hat Integration 2021.Q4 Release Notes for Red Hat Integration 2021.Q4

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 3

CHAPTER 2. CAMEL EXTENSIONS FOR QUARKUS RELEASE NOTES 4
2.1. CAMEL EXTENSIONS FOR QUARKUS FEATURES 4
2.2. SUPPORTED PLATFORMS, CONFIGURATIONS, DATABASES, AND EXTENSIONS 4
2.3. TECHNOLOGY PREVIEW EXTENSIONS 4
2.4. KNOWN ISSUES 4
2.5. IMPORTANT NOTES 5
2.6. ADDITIONAL RESOURCES 6

CHAPTER 3. CAMEL K RELEASE NOTES 7
3.1. NEW CAMEL K FEATURES 7
3.2. SUPPORTED CONFIGURATIONS 7
3.2.1. Camel K Operator metadata 7
3.3. IMPORTANT NOTES 8
3.4. SUPPORTED CAMEL QUARKUS EXTENSIONS 8
3.4.1. Supported Camel Quarkus connector extensions 8
3.4.2. Supported Camel Quarkus dataformat extensions 9
3.4.3. Supported Camel Quarkus language extensions 10
3.4.4. Supported Camel K traits 10
3.5. SUPPORTED KAMELETS 11
3.6. CAMEL K KNOWN ISSUES 14
3.7. CAMEL K FIXED ISSUES 15
3.7.1. Enhancements in Camel K 1.6.0 15
3.7.2. Bugs resolved in Camel K 1.6.0 16

CHAPTER 4. RED HAT INTEGRATION OPERATORS 17
4.1. WHAT OPERATORS ARE 17
4.2. RED HAT INTEGRATION OPERATOR 17
4.2.1. Supported components 18
4.2.2. Support life cycle 19
4.2.3. Fixed issues 19
4.3. RED HAT INTEGRATION COMPONENT OPERATORS 19
4.3.1. 3scale Operators 19
4.3.2. AMQ Operators 19
4.3.3. Camel K Operator 20
4.3.4. Fuse Operators 20
4.3.5. Service Registry Operator 20

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 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

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 ofcamel-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.

2.5. IMPORTANT NOTES

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

A patched version of Camel Extensions for Quarkus (version 2.2.0-1) with a fix for the Log4j 2.x security issue, CVE-2021-44228 (popularly known as Log4Shell) will shortly be made available through the Quarkus Platform. In the meantime, users of Camel Quarkus are unaffected by Log4Shell unless one of the following two conditions applies:

1. **camel-quarkus-corda** or **camel-quarkus-nsq** is used in the application. These extensions transitively depend on **log4j-core** but they do not require it to work properly on Quarkus. This is because JBoss log-manager is used as a logging backend on Quarkus. Hence the best mitigation is to exclude **log4j-core** from those dependencies:

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-nsq</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-core</artifactId>
    </exclusion>
  </exclusions>
</dependency>

<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-corda</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-core</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

   Note that **camel-quarkus-corda** or **camel-quarkus-nsq** are **not supported** by Red Hat and, therefore, use of these extensions is at your own risk.

2. Your application depends on **org.apache.logging.log4j:log4j-core** directly. In this case, make sure you upgrade to the newest log4j version (2.16.0 at the time of writing), where the Log4Shell vulnerability is fixed.

Documentation availability

The documentation set for *Red Hat build of Quarkus version 2.2* currently does not contain the full set of guides that we provide with releases of *Red Hat build of Quarkus*, so this documentation set for

Camel Extensions for Quarkus contains links to the *Red Hat build of Quarkus 1.11* documentation instead.

2.6. ADDITIONAL RESOURCES

- Supported Configurations
- Camel Extensions for Quarkus
- Getting Started with Camel Extensions for Quarkus
- Developing Applications with Camel Extensions for Quarkus

CHAPTER 3. 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.

3.1. 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

3.2. 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

3.2.1. 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 .

3.3. IMPORTANT NOTES

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

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 **1.6.x** 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, but the error messaging for incorrect YAML DSL code is still in development.

JAVA DSL Limitations

Java DSL in Camel K 1.6 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.

Camel K 1.6 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.

3.4. 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).

3.4.1. 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**
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**

3.4.2. 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**

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

3.4.3. 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

3.4.4. 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

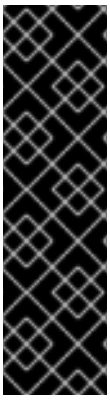
- Pull Secret trait
- Prometheus trait
- Quarkus trait
- Route trait
- Service trait
- Error Handler trait

3.5. 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 3.1. 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

Kamelet	File name	Type (Sink, Source, Action)
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
Elasticsearch Index sink*	**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)

Kamelet	File name	Type (Sink, Source, Action)
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)
Log sink	**log-sink.kamelet.yaml**	Sink (for development and testing purposes)
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

Kamelet	File name	Type (Sink, Source, Action)
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)
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*	**telegram-source.kamelet.yaml**	Source
Timer source	**timer-source.kamelet.yaml**	Source (for development and testing purposes)
TimeStamp Router action	**timestamp-router-action.kamelet.yaml**	Action (router)
Value to Key action	**value-to-key-action.kamelet.yaml**	Action (data transformation)

3.6. CAMEL K KNOWN ISSUES

The following known issues apply to the Camel K 1.6:

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.

3.7. CAMEL K FIXED ISSUES

The following sections list the issues that have been fixed in Camel K 1.6.0.

- Section 3.7.1, "Enhancements in Camel K 1.6.0"
- Section 3.7.2, "Bugs resolved in Camel K 1.6.0"

3.7.1. Enhancements in Camel K 1.6.0

The following table lists the enhancements in Camel K 1.6.0.

Table 3.2. Camel K 1.6.0 Enhancements

Issue	Description
ENTESB-17174	Support Timer and Log Kamelets (for development purposes)
ENTESB-17243	Remove usage of deprecated extensions/v1beta1 Ingress in Camel K

3.7.2. Bugs resolved in Camel K 1.6.0

The following table lists the resolved bugs in Camel K 1.6.0.

Table 3.3. Camel K 1.6.0 Resolved Bugs

Issue	Description
ENTESB-17974	CVE-2021-44228 log4j-core: Remote code execution in Log4j 2.x when logs contain an attacker-controlled string value [rhint-camel-k-1]
ENTESB-17626	Unrecognized field "firstTruthyTime" in Event Streaming Example
ENTESB-17412	Backport CAMEL-17039 - Camel-AWS2-S3: When includeBody is false, the message Body should not be set
ENTESB-17177	Operator 1.4 not working well with AMQ-Streams Operator 1.8
ENTESB-16733	OCP Developer console EventSource catalog contains Sink Kamelets
ENTESB-17123	Quickstart Camel K: Event Streaming Example warns about Secret in UserReportSystem integration
ENTESB-17129	bad kamelet resolution using global flag
ENTESB-17334	AWS Cloud Watch Kamelet: Header mapping is wrong
ENTESB-17174	Camel K APIs are marked as TechPreview
ENTESB-17260	Kbind resolves "channel/messages" as Kamelet "messages" in namespace "channel"
ENTESB-17254	camel-k-example-knative - endpoint has been deprecated
ENTESB-17314	Telegram-source seems to not emit proper cloud-events
ENTESB-17257	Kbind requires property "apiVersion" to create KameletBinding with InMemoryChannel
ENTESB-17004	Kamelets in namespace are not updated with the changes from operator
ENTESB-17835	Operator installed through OLM doesn't build integrations

CHAPTER 4. 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.

4.1. 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.

4.2. RED HAT INTEGRATION OPERATOR

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

4.2.1. 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.

4.2.2. 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.

4.2.3. 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.

4.3. 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.

4.3.1. 3scale Operators

- 3scale Operator
- 3scale APIcast Operator

4.3.2. AMQ Operators

- AMQ Broker Operator

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

4.3.3. Camel K Operator

- Camel K Operator - Technology Preview

4.3.4. Fuse Operators

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

4.3.5. 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.