Red Hat Integration 2020-Q4

Release Notes for Red Hat Integration 2020-Q4

What's new in Red Hat Integration

Last Updated: 2021-03-15

Red Hat Integration 2020-Q4 Release Notes for Red Hat Integration 2020-Q4

What's new in Red Hat Integration

Legal Notice

Copyright © 2021 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

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:

- API connectivity
- Data transformation
- Service composition and orchestration
- Real-time messaging
- Cross-datacenter message streaming
- API management

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 2020-Q4 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 2020-Q4 only. For details on the latest versions of other components, such as Debezium and Data Virtualization, see Red Hat Integration Release Notes for 2020-Q3 .

2.1. NEW INTEGRATION FEATURES

Serverless Camel K

- Cloud-native integration for serverless architectures with Red Hat Integration - Camel K (Technology Preview)
- Camel components available as connectors in Kafka Connect with Camel Kafka Connector (Technology Preview)

Apache Kafka

- Improved Kafka schema registry and OpenShift Operator with Service Registry 1.1

2.2. NEW COMPONENT FEATURES

For more details on what's new in Red Hat Integration 2020-Q4 components:

- Red Hat 3scale API Management
  - Red Hat 3scale API Management 2.9 On-Premises Release Notes
  - Red Hat 3scale API Management SaaS Release Notes
- Red Hat AMQ 2020-Q4 Product Documentation
- Red Hat Fuse 7.8 Release Notes

CHAPTER 3. CAMEL K RELEASE NOTES

Red Hat Integration - Camel K is available as a Technology Preview component in Red Hat Integration 2020-Q4. 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.

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. NEW CAMEL K FEATURES

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

3.1.1. Platform and core component versions

- OpenShift Continer Platform 4.6
- OpenShift Serverless 1.7
- Quarkus 1.7 Java runtime
- Camel 3.5
- Java 11

3.1.2. Technology Preview 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, XML, or YAML DSL
- Development tooling with Visual Studio Code

- Monitoring of integrations using Prometheus in OpenShift
- Quickstart tutorials, including new Transformations and SaaS
- Kamelet Catalog for source connectors to external systems such as AWS, Jira, and Salesforce

Additional resources

- Deploying Camel K Integrations on OpenShift

3.1.3. Camel K Operator metadata

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

Additional resources

- Operator bundle format in the OpenShift documentation .

3.2. CAMEL K KNOWN ISSUES

The following known issues apply to the Camel K Technology Preview:

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 -
```

CHAPTER 4. CAMEL KAFKA CONNECTOR RELEASE NOTES

Camel Kafka Connector is available as a Technology Preview component in Red Hat Integration 2020-Q4. Using Camel Kafka Connector, you can configure standard Camel components as connectors in Kafka Connect. This widens the scope of possible integrations beyond the external systems supported by Kafka Connect alone.

Camel Kafka Connector provides a user-friendly way to configure Camel components directly in the Kafka Connect framework. You can leverage Camel components for integration with different systems by connecting to or from Camel Kafka sink or source connectors. You do not need to write any code, and can include the appropriate connector JARs in your Kafka Connect image and configure the connector options using custom resources.

Camel Kafka Connector is built on Apache Camel Kafka Connector, which is a subproject of the Apache Camel open source community. Camel Kafka Connector is fully integrated with OpenShift Container Platform, AMQ Streams, and Kafka Connect. Camel Kafka Connector is available with the Red Hat Integration - Camel K distribution for cloud-native integration on OpenShift.

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.

4.1. NEW CAMEL KAFKA CONNECTOR FEATURES

The Camel Kafka Connector Technology Preview includes the following main features:

4.1.1. Platform and core component versions

- OpenShift Container Platform 4.5 or 4.6
- Red Hat Enterprise Linux 7.x, 8.x
- AMQ Streams 1.5
- Kafka Connect 2.5
- Camel 3.5

4.1.2. Technology Preview features

- Selected Camel Kafka connectors
- Marshaling/unmarshalling of Camel data formats for sink and source connectors
- Aggregation for sink connectors
- Maven archetypes for extending connectors

4.1.3. Technology Preview Camel Kafka connectors

Table 4.1. Camel Kafka connectors in Technology Preview

Connector	Sink/source
Amazon Web Services (AWS2) Kinesis	Sink and source
Amazon Web Services (AWS2) S3	Sink and source
Amazon Web Services (AWS2) SNS	Sink only
Amazon Web Services (AWS2) SQS	Sink and source
Cassandra Query Language (CQL)	Sink and source
Elasticsearch	Sink only
File	Sink only
Hadoop Distributed File System (HDFS)	Sink only
Hypertext Transfer Protocol (HTTP)	Sink only
Java Database Connectivity (JDBC)	Sink only
Java Message Service (JMS)	Sink and source
MongoDB	Sink and source
Salesforce	Source only
Slack	Source only
Syslog	Source only
Timer	Source only

Additional resources

- Getting Started with Camel Kafka Connector

CHAPTER 5. SERVICE REGISTRY RELEASE NOTES

Red Hat Integration - Service Registry 1.1 is provided as a General Availability release in Red Hat Integration 2020-Q4. 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 REST interface. For example, client applications can dynamically push or pull the latest 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.

5.1. SERVICE REGISTRY INSTALLATION OPTIONS

You can install Service Registry with the following storage options:

Table 5.1. Service Registry storage options

Storage option	Release
Kafka Streams-based storage in AMQ Streams 1.6	General Availability
Cache-based storage in embedded Infinispan 10	Technology Preview only
Java Persistence API-based storage in PostgreSQL 12 database	Technology Preview only

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.

5.2. SERVICE REGISTRY NEW FEATURES

Service Registry 1.1 includes the following updates:

Service Registry REST client

- Replaced the generated REST client with a new client based on Retrofit. This has fewer dependencies and is easier to package and use.
- REST client now supports TLS authentication.
- HTTP basic authentication credentials can be encoded in the registry URL or configured in REST client properties.

Service Registry serializers/deserializers (serdes)

- Avro serdes now supports JSON encoding - previously supported binary encoding only.
- Avro serdes updated with option to use message headers to pass **GlobalId** information instead of magic byte payload approach.
- Serdes layer now includes convenience wrapper classes for Kafka Streams applications, for example, wraps the serializer/deserializer pair in **AvroSerde** wrapper class.
- You can now configure custom HTTP request headers as properties when using serdes.

Service Registry configuration

- New environment variables to configure:
  - URLs for Service Registry web console and REST API - also improved the logic that generates these URLs when not set as variables.
  - Kafka topics for Kafka Streams storage.
  - Default global content rules - you can still override these using the **/rules** API.
- New **properties** field for artifact and version metadata to configure arbitrary key/value metadata on artifacts.

Service Registry management

- Registry Maven plug-in now chooses a default file extension based on the artifact type rather than defaulting to the Avro (**.avsc**) extension.
- Loosened restrictions on state changes, for example, **Deprecated** artifacts can now be undeprecated by transitioning back to **Enabled** state.
- Added metrics for HTTP Response status codes.

Service Registry web console

- Added a **Format** button to the **Content** tab for users to format JSON-encoded artifact content for easier readability.
- Added a tag to visually distinguish **Deprecated** and **Disabled** artifacts.

Service Registry Operator

- Operator now creates **ServiceMonitor** for Operator metrics.
- Operator metadata image is now available.
- Added **x-descriptors** to describe fields in OpenShift web console.
- Improved Operator label management.

Service Registry user documentation and examples

- Comprehensive details on using Kafka client serializers/deserializers for Avro, JSON schema, and Protobuf.
- Improved Service Registry Operator user documentation.

- New example applications available from https://github.com/Apicurio/apicurio-registry-examples.
  For more details, see:
  - Getting Started with Service Registry
  - Registry REST API documentation

5.3. SERVICE REGISTRY RESOLVED ISSUES

The following issues were resolved in Service Registry 1.1:

5.3.1. Service Registry core resolved issues

Registry-703 - Quarkus HTTP port configuration

Fixed error in HTTP port configuration for Quarkus Java runtime.

Registry-717 Registry allows empty artifact content

Registry no longer allows empty artifact content, which is now rejected at the API level.

Registry-737 - USE_SPECIFIC_AVRO_READER parameter

The **USE_SPECIFIC_AVRO_READER** configuration parameter now works as designed and no longer requires another parameter to also be set.

Registry-820 Cannot disable Compatibility rule

Added **NONE** option to the compatibility level configuration to allow explicit disabling of **Compatibility** rules at the artifact granularity.

Registry-853 Deprecated and Disabled artifacts in the web console

Various fixes related to handling of **Deprecated** and **Disabled** artifacts in the web console.

IBM compatibility REST API

Various fixes to the IBM compatibility REST API have been contributed by IBM.

5.3.2. Service Registry Operator resolved issues

Operator-26 - Error running Operator locally

Fixed client issue when running the Service Registry Operator locally. See also Operator-29

Operator-45 - Operator might not delete all resources

The Service Registry Operator now cleans up all resources when deleting the **ApicurioRegistry** custom resource definition. See also Operator-43

IPT-177 - Service Registry failed Kafka authentication

Service Registry no longer fails Kafka authentication if Salted Challenge Response Authentication Mechanism (SCRAM) password contains a **@** character.

5.4. SERVICE REGISTRY KNOWN ISSUES

The following known issues apply to Service Registry 1.1:

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-41 - Example CRD should not be empty

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

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.

5.5. SERVICE REGISTRY FUTURE CHANGES

We plan to make changes in an upcoming Service Registry release in response to user feedback about required functionality. Some breaking API changes will be required in the next Service Registry release to deliver the requested changes.

Because of these API changes, we will designate the next Service Registry release as version 2.0 instead of a standard version 1.x minor release. Both the current version 1.1 and the future version 2.0 will be supported in parallel for some time to give users a reasonable amount of time to migrate to version 2.0.

CHAPTER 6. RED HAT INTEGRATION OPERATORS

Red Hat Integration provides Operators to enable you to automate the deployment of Red Hat Integration components on OpenShift. This section provides links to detailed information on how to use Operators for components in this release.

6.1. 3SCALE OPERATOR

- 3scale Operator

6.2. AMQ OPERATORS

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

6.3. CAMEL K OPERATOR

- Camel K Operator (Technology Preview)

6.4. FUSE OPERATORS

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

6.5. SERVICE REGISTRY OPERATOR

- Service Registry Operator

Additional resources

- Understanding Operators in the OpenShift documentation
- OpenShift tech topic on Operators