Red Hat Integration 2020-Q2

Release Notes for Red Hat Integration 2020-Q2

What's new in Red Hat Integration

Last Updated: 2020-08-21

Red Hat Integration 2020-Q2 Release Notes for Red Hat Integration 2020-Q2

What's new in Red Hat Integration

Legal Notice

Abstract

Describes the Red Hat Integration platform and provides the latest details on what's new in this release.

Table of Contents

CHAPTER 1. RED HAT INTEGRATION

Red Hat Integration is a unified platform for cloud-native integration and application development with end-to-end API lifecycle support. This platform provides a set of agile and flexible integration and messaging technologies that include:

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

Red Hat Integration connects applications, data, and devices across hybrid cloud architectures and delivers API-centric business services.

CHAPTER 2. NEW FEATURES

This section provides a summary of the key new features in Red Hat Integration 2020-Q2 and provides links to more detailed information on new features available in different components.

2.1. KEY FEATURES

Serverless Camel K

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

Apache Kafka

- New OpenShift Operator and user interface for Kafka schema registry using Service Registry 1.0 General Availability
- Improved data replication across Kafka clusters using MirrorMaker 2.0 in AMQ Streams 1.5

Data integration

- Change data capture and real-time events with Debezium 1.1
- Container-native and API-based database access with Data Virtualization (Technology Preview)

2.2. NEW COMPONENT FEATURES

For more details on what's new in Red Hat Integration 2020-Q2 components:

- Red Hat AMQ 7.7 Product Documentation
- Red Hat Fuse 7.7 Release Notes
- AMQ Streams 1.5 on OpenShift Release Notes

CHAPTER 3. CAMEL K

Red Hat Integration - Camel K is available as a Technology Preview feature in Red Hat Integration 2020-Q2. 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. CAMEL K FEATURES

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

- OpenShift Continer Platform 4.3
- OpenShift Serverless 1.7
- Knative Serving for autoscaling and scale-to-zero
- Knative Eventing for event-driven architectures
- Java 11
- Performance optimizations using Quarkus 1.3 Java runtime
- Camel integrations written in Java, XML, or YAML DSL
- Development tooling with Visual Studio Code

Additional resources

- Deploying Camel K Integrations on OpenShift

CHAPTER 4. CAMEL KAFKA CONNECTOR

Camel Kafka Connector is available as a Technology Preview feature in Red Hat Integration 2020-Q2. 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. CAMEL KAFKA CONNECTOR FEATURES

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

- OpenShift Container Platform 4.4 or 4.3
- AMQ Streams 1.5
- Kafka Connect 2.5
- Camel 3.1
- Selected Camel Kafka connectors

4.2. AVAILABLE CAMEL KAFKA CONNECTORS

Table 4.1. Camel Kafka connectors in Technology Preview

Connector	Sink/source
Amazon AWS Kinesis	Sink and source
Amazon AWS S3	Sink and source

Connector	Sink/source
Cassandra Query Language (CQL)	Sink only
Elasticsearch	Sink only
Java Message Service (JMS)	Sink and source
Salesforce	Source only
Syslog	Source only

Additional resources

- Getting Started with Camel Kafka Connector

CHAPTER 5. DEBEZIUM

Red Hat Integration 2020-Q2 includes a General Availability release of Debezium on OpenShift based on the Debezium open source project. Debezium is a distributed change data capture platform that tracks database operations and streams change data events. Debezium is built on Apache Kafka and is deployed and integrated with AMQ Streams.

Debezium captures row-level changes to database tables and passes corresponding change event records to AMQ Streams. Applications can read these *change event streams* and access the change events in the order in which they occurred.

The following topics provide release details:

5.1. DEBEZIUM DATABASE CONNECTORS

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

- **MySQL**
- **PostgreSQL**
  The Debezium 1.1.2 version of the PostgreSQL connector, which was originally part of Red Hat Integration 2020-Q2, has a security vulnerability. See Debezium release 1.1.3 for details about the PostgreSQL connector update in Debezium 1.1.3, which is now part of Red Hat Integration 2020-Q2, and which fixes this vulnerability.
- **MongoDB** - Supported in this release. It was a Technology Preview feature in the previous release.
- **SQL Server** - Supported in this release. It was a Technology Preview feature in the previous release.

5.2. SUPPORTED DATABASE VERSIONS FOR DEBEZIUM

When trying out the database connectors, the following database versions are supported for this release:

Database	Versions
MySQL	5.7, 8.0
PostgreSQL	10, 11, 12
MongoDB	3.6, 4.0, 4.2

Database	Versions
SQL Server	2017, 2019

NOTE

For PostgreSQL deployments, you use the **pgoutput** logical decoding output plug-in, which is the default for PostgreSQL versions 10 and later.

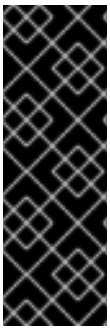
Additional resources

- Getting Started with Debezium
- Debezium User Guide

5.3. DEBEZIUM INSTALLATION OPTIONS

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

- Installing Debezium on OpenShift - General Availability
- Installing Debezium on RHEL - Technology Preview



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.

5.4. NEW DEBEZIUM FEATURES

This release provides the following new Debezium features:

- **ByLogicalTableRouter** single message transformation for re-routing data change event records to topics that you specify.
- **ExtractNewRecordState** single message transformation for flattening the complex structure of a data change event record into a simplified format that some Kafka Connect consumers might require.
- **CloudEventsConverter** for emitting change event records that conform to the CloudEvents specification. This is a Technology Preview feature.
- The MongoDB, PostgreSQL, and SQL Server connectors support **transaction metadata**. When using these connector types, Debezium can generate events that represent transaction metadata boundaries. This lets downstream consumers group and aggregate change data event records that originate from an individual transaction.

When using the Debezium connector for MongoDB, there is a limitation if you are using MongoDB 4.2. The limitation is that you cannot use the connector's transaction metadata feature. This limitation is expected to be removed in a future release.

- **Avro serialization** - You can configure Debezium connectors to use Avro to serialize message keys and values. This is a Technology Preview feature.

5.5. DEBEZIUM 1.1.3 RELEASE

Debezium 1.1.3 provides an updated PostgreSQL connector. There are no other updates in Debezium 1.1.3.

Red Hat Integration 2020-Q2 originally included the Debezium 1.1.2 release and now includes the Debezium 1.1.3 release. The 1.1.3 release updates only the PostgreSQL connector. The update to the connector is that it now uses version 42.2.14 of the PostgreSQL JDBC driver, which fixes a Common Vulnerability and Exposure (CVE-2020-13692) issue. Consequently, if you are already using a release of the Debezium PostgreSQL connector, it is recommended that you upgrade to the PostgreSQL connector provided in Debezium 1.1.3. To do this, go to the Red Hat Integration 2020-Q2 Software Downloads, Security Advisories tab and download the Debezium 1.1.3 PostgreSQL Connector.

Additional details are available in the Debezium 1.1x Resolved Issues knowledge article .

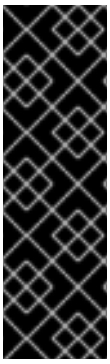
The use of the newer JDBC driver changes PostgreSQL connector snapshot behavior for data change event records for partitioned tables. Previously, the connector sent snapshot change event records to a topic that corresponded to the parent table. In this release, the connector routes snapshot change event records for a partitioned table to a different topic for each partition. If you need to change this behavior, you can configure the **ByLogicalTableRouter** single message transformation (SMT), which is a new feature in Red Hat Integration 2020-Q2.

The CVE that was fixed permitted an XML external entity attack (see https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing). When processing XML contents from untrusted sources with the older version of the PostgreSQL JDBC driver, an attack could lead to "the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts".

CHAPTER 6. DATA VIRTUALIZATION

Data Virtualization is available as a Technology Preview feature in Red Hat Integration 2020-Q2. Data Virtualization is a container-native service, based on the Teiid open source project, that provides integrated access to a range of data sources, including relational databases, MongoDB, and Salesforce, through a single uniform API.

Applications and users connect to a virtual database over standard interfaces (OData REST, or JDBC/ODBC) and can use SQL to interact with the integrated data in the same way that they would interact with a single relational database.



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.

6.1. DATA VIRTUALIZATION ENHANCEMENTS

Red Hat Integration 2020-Q2 provides the following enhancements for Data Virtualization:

- Simplified format for defining data sources in the custom resource for a virtual database.
- Compatibility with the following additional data sources:
  - Ceph
  - SFTP
  - SOAP
- Ability to import from existing virtual databases.
- Custom Prometheus metrics.
- Ability to configure a load balancer for JDBC/ODBC traffic.
- Jaeger integration.
- Improved handling of TLS certificates for encrypting connections from clients and to data sources.
- Ability to configure a custom Maven settings file.
- Passing down proxy settings to operands.
- New user documentation:
  - Data Virtualization Tutorial

6.2. DATA VIRTUALIZATION KNOWN ISSUES

ENTESB-14311 - Data Grid based materialization does not work

Materialized views that are stored in the Data Grid cache can not be configured in this release. When a virtual database is deployed into a namespace where the Data Grid or Infinispan Operators are present, and a virtual database has configured a view with materialization (for example, OPTIONS (MATERIALIZED 'TRUE')), the Data Virtualization Operator automatically creates a Data Grid cache store, for the purposes of materialization. However, after the virtual database is deployed, queries that you submit to fetch the contents of the view return the following error:

```
TEIID30384 Error while evaluating function mvstatus
```

To resolve the error, disable use of the Data Grid or Infinispan cache, and revert to using the default materialization mechanism by completing the following steps.

1. Set the environment property **DISABLE_ISPN_CACHING** in the CR to **true**. For example,

```
apiVersion: teiid.io/v1alpha1
kind: VirtualDatabase
metadata:
  name: matexample
spec:
  replicas: 1
  env:
    - name: DISABLE_ISPN_CACHING
      value: "true"
  build:
    source:
 ....
```

2. Run the Data Virtualization Operator with the updated CR.
   After you build the virtual database with the specified environment variable, the caching mechanism reverts to the default method.

ENTESB-14297 - Connecting to SAP HANA does not work with the prescribed driver

When you create a virtual database that uses SAP HANA as a data source, the JDBC driver that is specified as a build dependency for the virtual database is unable to connect to HANA. Tested using SAP HANA version 1.00.102.

ENTESB-13462 - FORMAT functions for different versions of the Data Virtualization Operator return different results

When you use format functions such as **FORMATBIGDECIMAL** or **FORMATDOUBLE** to query a virtual database, the results that the functions return for negative numbers might not be formatted as expected. The format of the result depends on the underlying Java version specified by the Data Virtualization Operator. Because recent versions of the Operator switched from using Java 8 to Java 11, you might experience changes in the formatting of your query results.

In Java versions earlier than Java 11, regardless of whether you specify a minus sign character (**-**) as a prefix or suffix of a negative format pattern, the returned negative result string is preceded by a minus sign. However, in Java 11, if you append a minus sign to a negative format pattern, the result string that is returned has the minus sign appended to it.

If you obtain unexpected results for negative numbers that a parsing or formatting function returns, revise the pattern string for the function so that the minus sign precedes the rest of the pattern.

Example

The following query specifies the negative format pattern, **#,0;#0-**.

```
SELECT FORMATBIGDECIMAL((99 - 10000000), '#,##0;###0-')
```

Depending on the version of the Operator, one of the following results is returned:

DV Operators using Java 8

-9,999,901

DV Operators using Java 11

9,999,901-

To achieve the same result as with Java 8, modify the formatting pattern for the query to **,#0;-#**.

6.3. DATA VIRTUALIZATION MIGRATION CONSIDERATIONS

You can migrate virtual database that you developed for JBoss Data Virtualization or Teiid deployments. For information about migrating legacy virtual databases to Data Virtualization, see Using Data Virtualization.

CHAPTER 7. SERVICE REGISTRY

Red Hat Integration - Service Registry 1.0 is provided in a General Availability release in Red Hat Integration 2020-Q2. Service Registry is a datastore for standard event schemas and API designs that 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 type validation or backwards and forwards compatibility of content versions.

7.1. SERVICE REGISTRY INSTALLATION OPTIONS

You can install Service Registry with the following storage options:

Table 7.1. Service Registry storage options

Storage option	Release
Kafka Streams-based storage in AMQ Streams 1.5	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.

7.2. NEW SERVICE REGISTRY FEATURES

Service Registry provides the following new features in this 1.0 General Availabilty release:

Service Registry OpenShift Operator

The new Red Hat Integration - Service Registry Operator is available from the OpenShift OperatorHub for installation and upgrade. This replaces the OpenShift template previously available in Technology Preview versions. Example custom resource definitions are also provided for storage in AMQ Streams, embedded Infinispan, or PostgreSQL database.

Service Registry web console

- Search and browse for artifacts and versions (based on new Search REST API)
- View details about an artifact, and view generated documentation (OpenAPI only)

- Perform artifact actions:
  - Upload to registry
  - Edit metadata
  - Configure rules
  - Download locally
- Configure global rules

Additional storage options (Technology Preview only)

- Java Persistence API (JPA) - PostgreSQL 12.x database
- Embedded Infinispan cache - Infinispan 10.0.1.Final (Community build)

Additional artifact types

- WSDL - Web Services Description Language
- XSD - XML Schema Definition
- XML - Extensible Markup Language

Improved rules for registry content

- Improved compatibility rule for JSON Schema artifacts

Improved artifact metadata

- Automatic extraction of metadata defined in artifact content, for example, artifact name and description
- Labels for searching and browsing

Quarkus version

- Red Hat build of Quarkus 1.3.2.Final

New user documentation

- Getting Started with Service Registry
- Registry REST API documentation
- Creating Service Registry client applications and serializers/deserializers with AMQ Streams
- Deploying a Debezium connector and Apache Avro serialization with Service Registry

7.3. SERVICE REGISTRY KNOWN ISSUES

Service Registry core known issues

IPT-168 - Cannot compile project using released client Serde artifact

Some dependencies that are required for the Service Registry client serializer/deserializer Java classes are missing from the Red Hat GA Maven repository.

The following workaround options are available:

- Manually exclude or override the missing dependencies
- Import the **quarkus-bom** as a dependency in your **pom.xml** file:

```
<quarkus.version>1.3.4.Final-redhat-00004</quarkus.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <!-- Quarkus Dependencies -->
      <dependency>
        <groupId>io.quarkus</groupId>
        <artifactId>quarkus-bom</artifactId>
        <version>${quarkus.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
```

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-41 - Example CRD should not be empty::

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.

Operator-45 - Operator may not delete all resources

The Service Registry Operator may not delete all resources when deleting **ApicurioRegistry** custom resource definition.

CHAPTER 8. AMQ STREAMS

AMQ Streams simplifies the process of running Apache Kafka in an OpenShift cluster.

8.1. MIRRORMAKER 2.0

Support for MirrorMaker 2.0 moves out of Technology Preview to a generally available component of AMQ Streams.

MirrorMaker 2.0 is based on the Kafka Connect framework, with *connectors* managing the transfer of data between clusters.

MirrorMaker 2.0 uses:

- Source cluster configuration to consume data from the source cluster
- Target cluster configuration to output data to the target cluster

MirrorMaker 2.0 introduces an entirely new way of replicating data in clusters. If you choose to use MirrorMaker 2.0, there is currently no legacy support, so any resources must be manually converted into the new format.

For more information, see MirrorMaker 2.0 in AMQ Streams 1.5 .

CHAPTER 9. 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.

9.1. AMQ OPERATORS

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

9.2. CAMEL K OPERATOR

- Camel K Operator (Technology Preview)

9.3. DATA VIRTUALIZATION OPERATOR

- Data Virtualization Operator (Technology Preview)

9.4. FUSE OPERATORS

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

9.5. SERVICE REGISTRY OPERATOR

- Service Registry Operator

Additional resources

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