Red Hat Integration 2023.q4

Release Notes for Red Hat Integration 2023.q4

What's new in Red Hat Integration

Last Updated: 2024-03-22

Red Hat Integration 2023.q4 Release Notes for Red Hat Integration 2023.q4

What's new in Red Hat Integration

Abstract

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

Table of Contents

PREFACE 4
MAKING OPEN SOURCE MORE INCLUSIVE 4

CHAPTER 1. RED HAT INTEGRATION 5

CHAPTER 2. DEBEZIUM 2.3.4 RELEASE NOTES 6
2.1. DEBEZIUM DATABASE CONNECTORS 6
2.1.1. Connector usage notes 6
2.2. DEBEZIUM SUPPORTED CONFIGURATIONS 7
2.2.1. AMQ Streams API version 7
2.3. DEBEZIUM INSTALLATION OPTIONS 7
2.4. UPGRADING DEBEZIUM FROM VERSION 1.X TO 2.3.4 7
2.4.1. Upgrading connectors to Debezium 2.3.4 7
2.5. NEW FEATURES AND IMPROVEMENTS 8
2.5.1. Breaking changes 9
2.5.1.1. New configuration defaults for MySQL and PostgreSQL secure connections 9
2.5.1.2. Topic and schema naming changes 9
2.5.1.3. Changes to the Oracle connector source information block 10
2.5.2. Features promoted to General Availability 10
2.5.3. General availability features 10
2.5.3.1. Automated replica identity configuration for PostgreSQL 11
2.5.3.2. New notification subsystem 11
2.5.3.3. Correlate incremental snapshot notification IDs 12
2.5.3.4. Support for new signaling channels 12
2.5.3.5. Oracle RAC improvements 13
2.5.3.6. Oracle connector SCN-based metrics 13
2.5.3.7. Server side filtering for the MongoDB and Oracle connectors 13
2.5.3.8. Retry database connections during connector startup 14
2.5.3.9. Use of surrogate keys in incremental snapshots 14
2.5.3.10. ExtractChangedRecordState SMT 15
2.5.3.11. Drop event fields with new configuration options for the ExtractNewRecordState SMT 16
2.5.3.12. HeaderToValue SMT 17
2.5.3.13. Partition routing SMT 17
2.5.4. Technology Preview features 17
2.5.4.1. MongoDB sharded cluster improvements (Technology Preview) 17
2.5.4.2. MongoDB incremental snapshots for multi-replica and sharded clusters (Technology Preview) 18
2.5.5. Developer Preview features 18
2.5.5.1. JDBC Sink Connector (Developer Preview) 19
2.5.5.2. MySQL connector: parallel snapshots (Developer Preview) 20
2.5.5.3. Oracle connector: Ingesting changes from an Oracle logical standby (Developer Preview) 20
2.5.5.4. Exactly-once delivery for PostgreSQL connector (Developer Preview) 20
2.5.6. Other updates in this release 21
2.6. DEPRECATED FEATURES 25
2.7. KNOWN ISSUES 25

CHAPTER 3. SERVICE REGISTRY 2.5 RELEASE NOTES 26
3.1. SERVICE REGISTRY INSTALLATION OPTIONS 26
3.2. SERVICE REGISTRY SUPPORTED PLATFORMS 26
3.2.1. Supported integration with other products 26
3.2.2. Operator metadata versions 27
3.3. SERVICE REGISTRY NEW FEATURES 27
Service Registry core new features 27

Service Registry Operator new features 28
Service Registry user documentation and examples 28
3.4. SERVICE REGISTRY DEPRECATED FEATURES 29
Service Registry core deprecated features 29
Service Registry Operator deprecated features 29
3.5. UPGRADING AND MIGRATING SERVICE REGISTRY DEPLOYMENTS 30
3.5.1. Updating 2.x client dependencies 30
3.5.2. Upgrading from Service Registry 2.x on OpenShift 30
3.5.3. Migrating from Service Registry 1.1 on OpenShift 31
3.6. SERVICE REGISTRY RESOLVED ISSUES 31
3.7. SERVICE REGISTRY RESOLVED CVES 32
3.8. SERVICE REGISTRY KNOWN ISSUES 36
Service Registry core known issues 36
Service Registry Operator known issues 36

APPENDIX A. USING YOUR SUBSCRIPTION **37**
ACCESSING YOUR ACCOUNT 37
ACTIVATING A SUBSCRIPTION 37
DOWNLOADING ZIP AND TAR FILES 37

PREFACE

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

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. DEBEZIUM 2.3.4 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 on OpenShift Container Platform or on Red Hat Enterprise Linux.

The following topics provide release details:

- Section 2.1, "Debezium database connectors"
- Section 2.2, "Debezium supported configurations"
- Section 2.3, "Debezium installation options"
- Section 2.4, "Upgrading Debezium from version 1.x to 2.3.4"
- Section 2.5, "New features and improvements"
- Section 2.6, "Deprecated features"
- Section 2.7, "Known issues"

2.1. DEBEZIUM DATABASE CONNECTORS

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

- Db2
- JDBC Sink connector (Developer preview)
- MongoDB
- MySQL
- Oracle
- PostgreSQL
- SQL Server

2.1.1. Connector usage notes

- Db2
  - The Debezium Db2 connector does not include the Db2 JDBC driver (**jcc-11.5.0.0.jar**). See the Db2 connector deployment instructions for information about how to deploy the necessary JDBC driver.
  - 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.

- Oracle
  - The Debezium Oracle connector does not include the Oracle JDBC driver (**ojdbc8.jar**). See the Oracle connector deployment instructions for information about how to deploy the necessary JDBC driver.
- PostgreSQL
  - To use the Debezium PostgreSQL connector you must 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

2.2. DEBEZIUM SUPPORTED CONFIGURATIONS

For information about Debezium supported configurations, including information about supported database versions, see the Debezium 2.3.4 Supported configurations page .

2.2.1. AMQ Streams API version

Debezium runs on AMQ Streams 2.5.

AMQ Streams supports the **v1beta2** API version, which updates the schemas of the AMQ Streams custom resources. Older API versions are deprecated. After you upgrade to AMQ Streams 1.7, but before you upgrade to AMQ Streams 1.8 or later, you must upgrade your custom resources to use API version **v1beta2**.

For more information, see the Debezium User Guide.

2.3. DEBEZIUM INSTALLATION OPTIONS

You can install Debezium with AMQ Streams on OpenShift or on Red Hat Enterprise Linux:

- Installing Debezium on OpenShift
- Installing Debezium on RHEL

2.4. UPGRADING DEBEZIUM FROM VERSION 1.X TO 2.3.4

The current version of Debezium includes changes that require you to follow specific steps when you upgrade from versions earlier than 2.1.4.

2.4.1. Upgrading connectors to Debezium 2.3.4

The Debezium 2.3.4 release includes some changes that are not backward-compatible with versions of Debezium earlier than 2.x. As a result, to preserve data and ensure continued operation when you upgrade from Debezium 1.x versions to 2.3.4, you must complete some manual steps during the upgrade process.

One significant change is that the names of some connector parameters have changed. To accommodate these changes, review the configuration properties updates in the 2023.Q2 release notes, and note the properties that are present in your connector configuration. Before you upgrade, edit the configuration of each Debezium connector to add the new names of any changed properties. Before you upgrade, edit the configuration of any 1.x connector instances so that both the old and new property names are present. After the upgrade, you can remove the old configuration options.

Prerequisites

- Debezium is now compatible with Kafka versions up to 3.5.0. This is the default Kafka version in AMQ Streams 2.5.
- The Java 11 runtime is required and must be available prior to upgrading. AMQ Streams 2.5 supports Java 11. Use Java 11 when developing new applications. Java 11 enables use of recent language updates, such as the new String API and changes in predicate support, while also benefiting from Java performance improvements. Java 8 is no longer supported in AMQ Streams 2.5.
- Check the backward-incompatible changes in the current list of breaking changes and in the 2023.Q2 release notes.
- Verify that your environment complies with the Debezium 2.3.4 Supported Configurations.

Procedure

1. From the OpenShift console, review the Kafka Connector YAML to identify the connector configuration that are no longer valid in Debezium 2.3.4. Refer to the 2023.Q2 release notes for details.
2. Edit the configuration to add the 2.x equivalents for the properties that you identify in Step 1, so that both the old and new property names are present. Set the values of the new properties to the values that were previously specified for the old properties.
3. From the OpenShift console, stop Kafka Connect to gracefully stop the connector.
4. From the OpenShift console, edit the Kafka Connect image YAML to reference the Debezium 2.3.4.Final version of the connector zip file.
5. From the OpenShift console, edit the Kafka Connector YAML to remove any configuration options that are no longer valid for your connector.
6. Adjust your application's storage dependencies, as needed, depending on the storage module implementation dependencies in your code. For more information, see Changes to Debezium storage in the 2023.Q2 release notes.
7. Restart Kafka Connect to start the connector. After you restart the connector, it continues to process events from the point where it stopped before the upgrade. Change events records that the connector wrote to Kafka before the upgrade are not modified.

2.5. NEW FEATURES AND IMPROVEMENTS

Debezium 2.3.4 includes the following updates and improvements:

- Breaking changes
- Features promoted to General availability

- General availability features
- Technology Preview features
- Developer Preview features
- Other updates in this release

2.5.1. Breaking changes

The following changes in Debezium 2.3.4 represent significant differences in connector behavior and require configuration changes that are not compatible with earlier Debezium versions: Debezium 2.3.4 introduces the following breaking changes:

- MySQL and PostgreSQL secure connection changes
- Topic and schema naming changes
- Source info block changed for Oracle connector

For information about breaking changes in the previous Debezium release, see the 2023.Q2 Release Notes.

2.5.1.1. New configuration defaults for MySQL and PostgreSQL secure connections

You can configure the Debezium connectors for MySQL and PostgreSQL to use secure SSL connections. For the MySQL connector, you specify use of a secure connection by configuring the **database.ssl.mode** property. For the PostgreSQL connector, you set the **database.sslmode** property.

Beginning with Debezium 2.3.4, these configuration options include new default values. For MySQL, the default value for **database.ssl.mode** is now **preferred**, replacing the previous default value of **disabled**. For PostgreSQL, the default value for **database.sslmode** is now **prefer**, replacing the previous default value of **disable**. Based on the new default settings, when the connectors initiate a connection to a database, they first attempt to establish an encrypted, secure connection. If a secure connection is not available, the connectors fall back to using an unsecured connection, unless configured otherwise.

2.5.1.2. Topic and schema naming changes

When Debezium generates topic names and schema names, it replaces non-ASCII characters in the names to ensure compatibility with the naming conventions of schema registries. In earlier releases, Debezium substituted an underscore character (_) to replace non-ASCII characters. However, in some cases, after replacing non-ASCII characters, the names that Debezium generates for two topics or schema, could be identical except for their letter casing, which could lead to other problems.

In order to address this in the most compatible way, Debezium now uses a strategy-based approach to map characters uniquely. One side effect of this new approach is that Debezium no longer supports the **sanitize.field.names** configuration property. In the place of the **sanitize.field.names** property, new options are now available for specifying a naming strategy that is compatible with the conventions that you use for your tables or collections.

To specify how Debezium generates schema and field names, you can set the following properties.

schema.name.adjustment.mode

Specifies how schema names should be adjusted for compatibility with the message converter.

field.name.adjustment.mode

Specifies how field names should be adjusted for compatibility with the message converter.

For each of the preceding properties, you can set one of the following values:

none

Names are passed as-is; no adjustments are made to schema or field names.

avro

Replaces characters that cannot be used in Avro with an underscore (_).

avro_unicode

Replaces underscores (_) and characters that cannot be used in Avro with unicode-based escape sequences.

2.5.1.3. Changes to the Oracle connector source information block

The change event record that Debezium emits for insert, update, and delete event includes a payload that contain a **source** information block. For the Oracle connector, the source information block contains a special **ssn** field that represents the SQL sequence number of a change.

In some cases, the value for the **ssn** field in the source database exceeds the maximum value of an **INT32** data type (**2,147,483,647**). To allow for larger values, Debezium now assigns the data type **INT64** to **ssn** fields, which increases the maximum value of the field to **9,223,372,036,854,775,807**.

If you currently store the **ssn** value in a sink system in your environment, or if you are using a schema registry, this change could affect the behavior of your system.

2.5.2. Features promoted to General Availability

The following features are promoted from Technology Preview to General Availability in the Debezium 2.3.4 release:

Ad hoc and incremental snapshots for MongoDB connector

Provides a mechanism for re-running a snapshot of a table for which you previously captured a snapshot.

Signaling for the MongoDB connector

Provides a mechanism for modifying the behavior of a connector, or triggering a one-time action, such as initiating an ad hoc snapshot of a table.

Content-based routing

Provides a mechanism for rerouting selected events to specific topics, based on the event content.

Filter SMT

Enables you to specify a subset of records that you want the connector to send to the broker.

2.5.3. General availability features

Debezium 2.3.4 supports the following new features:

- Automated replica identity configuration for PostgreSQL
- New notification subsystem (sink, log, JMX)
- Correlate incremental snapshot notification IDs

- Support for new signaling channels (source, Kafka, JMX)
- Oracle RAC improvements
- Oracle connector SCN-based metrics
- Server side filtering for MongoDB and Oracle
- Retry database connections on startup
- Surrogate keys for incremental snapshots
- ExtractChangedRecordState SMT
- **ExtractNewRecordState** (event flattening) SMT options for dropping fields from an event record
- HeaderToValue SMT
- Partition routing SMT

2.5.3.1. Automated replica identity configuration for PostgreSQL

Debezium 2.3.4 introduces a new PostgreSQL connector feature known as "Autoset Replica Identity".

A PostgreSQL database uses replica identity to identify the columns that are captured in the database transaction logs for insert, update, and delete events. This feature enables you to configure the connector to automatically set the replica identity value for a table. When the connector starts, it reads the replica identity configuration and then sets the replica identity for the specified tables.

The new configuration property, **replica.identity.autoset.values**, specifies a comma-separated list of table and replica identity tuples. When the property specifies a replica identity for a table, that value overrides any existing replica identity configuration. For more information about PostgreSQL replica identity types, see the PostgreSQL documentation.

The **replica.identity.autoset.values** property accepts a comma-separated list of values in which each element uses the format of <fully-qualified-table-name>:<replica-identity>. The following example shows how to configure two tables (**table1** and **table2**) to have **FULL** replica identity:

{ "replica.identity.autoset.values": "public.table1:FULL,public.table2:FULL" }

The user account through which the connector accesses the database requires permission to set the table’s replica identity. If the account lacks sufficient permissions, any attempt to use **replica.identity.autoset.values** results in a failure. If you cannot use the property to automatically set the replica identity, you must set the replica identity for the table manually, from a database account that has the required permission.

2.5.3.2. New notification subsystem

This release introduces a new notifications subsystem, which enables Debezium to emit events that report on the status of various connector operations, such as incremental or traditional snapshots. This new subsystem allows you to send a notification through several different channels, including Kafka topics, log files, and Java Management Extensions (JMX). These notification events can be consumed by a variety of external systems. Notification events are represented as a series of key/value tuples, including the following fields:

id

A UUID that identifies the notification,

aggregate_type

The type of notification, based on the concept of domain-driven design.

type

Provides more detail about the aggregate type.

additional_data (optional)

A map of string-based key/value pairs with additional information about the event.

The following example shows a simple notification event.

Example notification event

```
{
  "id": "c485ccc3-16ff-47cc-b4e8-b56a57c3bad2",
  "aggregate_type": "Snapshot",
  "type": "Started",
  "additional_data": {
    ...
  }
}
```

In this release, Debezium supports the following types of notification events:

- Status of the initial snapshot
- Incremental snapshot progress

For more information, see Configuring notifications to report connector status .

2.5.3.3. Correlate incremental snapshot notification IDs

In this release, the notification and channels subsystem has been improved to correlate the signal to the notification. That is, when you send a signal and it is consumed by Debezium, the resulting notification contains an identifier that references the original signal. When communications are distributed across multiple applications and processes, this mechanism enables processes to more easily associate signals with their resulting operations.

2.5.3.4. Support for new signaling channels

Debezium has supported signaling since the introduction of incremental snapshots in release 1.7. Signals provide a mechanism for using metadata to instruct Debezium to perform tasks, such as writing an entries to the connector log, or performing an ad-hoc incremental snapshot.

This release introduces support for multiple signaling channels, enabling you to specify the medium that Debezium uses to watch for and react to signals. In previous versions, there was one channel supported universally across connectors, which was the database signal table. In this release, the following are available by default:

- Database signal table
- Kafka signal topic
- JMX

In this release, the signal channel subsystem has been improved to support sending signals via JMX. From a JConsole window, two subsections now exist for a connector, a notifications section, and a signal section.

The **signal** section enables you to invoke an operation on a JMX bean to transmit a signal to Debezium. This signal resembles the logical signal table structure in that it accepts 3 parameters:

- A unique identifier
- The signal type
- The signal payload.

For more information, see Sending signals to a Integration connector

2.5.3.5. Oracle RAC improvements

When you use the Debezium Oracle connector with an Oracle Real Application Clusters (RAC) deployment, you must specify a **rac.nodes** configuration property. At minimum, the **rac.nodes** property must specify the host or IP address of each individual node in the cluster. Older versions of the connector also supported an alternate format in which you could specify a unique port number for each node, in recognition of the fact that different nodes might use different ports.

Debezium 2.3.4 improves Oracle RAC support by also recognizing that each node might use a different Oracle Site Identifier (SID). To account for variations in the Oracle SID configuration, you can now specify the SID paramter in the **rac.nodes** configuration property.

The following example illustrates connecting to two Oracle RAC nodes, each using different ports and SID parameters:

{ "connector.class": "io.debezium.connector.oracle.OracleConnector", "rac.nodes": "host1.domain.com:1521/ORCLSID1,host2.domain.com:1522/ORCLSID2", ... }

2.5.3.6. Oracle connector SCN-based metrics

Oracle tracks a variety of system change numbers (SCNs) values in its JMX metrics, including **OffsetScn**, **CurrentScn**, **OldestScn**, and **CommittedScn**. These SCN values are numeric and can often exceed the upper bounds of a **LONG** data type. In past releases, Debezium exposed SCN values as **String** values.

To improve the utility of these metrics, Debezium now exposes these JMX metrics as **BigInteger** values, rather than as **String** values. This change enables users to view values for these metrics through tools such as Grafana and Prometheus, which do not support string-based values.

NOTE

If you previously gathered SCN values for other purposes, be aware they are no longer string-based, and must be interpreted as **BigInteger** numerical values.

2.5.3.7. Server side filtering for the MongoDB and Oracle connectors

When fetching entries from the database, the MongoDB and Oracle connectors can now submit **include** and **exclude** filters that are set in the connector configuration to the database. The MongoDB connector does this automatically. If you want the Oracle connector to submit filters when fetching entries, set the **log.mining.query.filter.mode** property to a value other than **none**, which is the default.

In past releases, the MongoDB and Oracle connectors first fetched events from the database, and then evaluated events against the filter settings. This process effectively serialized all changes from the database across the network to the connector. an approach that is inefficient, especially in high-volume environments. Connectors received some events only to discard them immediately afterwards, due to filter settings. For connectors that run in cloud environments, transmitting such a large volume of excess data inflates utilization costs.

To reduce the amount of data that the connectors fetch, in Debezium 2.3.4, connectors no longer evaluate filters after fetching data. Instead, the include and exclude lists are defined in the MongoDB change stream subscription or the Oracle fetch query. By reducing the number of events that the connector reads, the new approach results in lower network and CPU utilization. For a connector that is configured with full document or pre-image settings, this adds even more utilization to the network that is entirely unnecessary. Furthermore, by enabling the connector to receive only events that require processing, the connector is able to complete more processing, raising CPU utilization.

2.5.3.8. Retry database connections during connector startup

In previous releases, connectors used a fail-fast strategy during startup. That is, if the connector could not perform any step required to complete the startup routine, for example, connect to the database or authenticate, the connector would enter a **FAILED** state.

In some situations, the connector might start gracefully, run for a period of time, and then eventually encounter a fatal error. Errors could be related to resources that were not accessed during the connector’s startup lifecycle, so that you could restart the connector without encountering an error. However, when a failure results because the database becomes unavailable, if the database remains unavailable after the connector restarts, the fail-fast strategy causes the connector to enter a **FAILED** state. You must then intervene manually to resolve the problem.

To improve reliability and resiliency, in this release, instead of attempting to access potentially unavailable resources during startup, the connector now attempts to access these resources later in its lifecycle. In effect, during startup, Debezium is less strict about accessing potentially unavailable resources, enabling it to take advantage of the Kafka Connect retry back-off framework.

Now, if a database is unavailable during connector startup, as long as Kafka Connect retries are enabled, the connector continues to retry failed requests. A **FAILED** state only results after the maximum number of retry attempts has been reached, or if a non-retriable error occurs.

2.5.3.9. Use of surrogate keys in incremental snapshots

The Debezium incremental snapshot feature provides a mechanism for performing resumable, consistent snapshots of data. This ability to resume snapshots can be critical for connectors that must ingest large volumes of data.

In earlier releases, incremental snapshots required that a primary key was set for every table included in the snapshot. Beginning with Debezium 2.3.4, you can now perform incremental snapshots on key-less tables, as long as the table includes one unique that can serve as a "surrogate key".

WARNING

The ability to use surrogate key for incremental snapshots applies only to the Debezium relational connectors; you cannot use this feature with the MongoDB connector.

To provide the surrogate key column data in an incremental snapshot signal, you must include the new surrogate key attribute, **surrogate-key** in the signal payload.

An example incremental snapshot signal payload specifying a surrogate key

```
{
  "data-collections": [ "public.mytab" ],
  "surrogate-key": "customer_ref"
}
```

The signal in the preceding example initiates an incremental snapshot for the table **public.mytab**. The snapshot uses the **customer_ref** column as the primary key for generating snapshot windows.

WARNING

You must use a single column to define a surrogate key. You cannot define surrogate keys that are based on multiple columns.

You can also use the surrogate key feature with tables that have primary keys. For example, surrogate keys offer an advantage when a table's primary key consists of multiple columns. Queries based on multiple columns generate a disjunction predicate for each column in the primary key, and the performance can be highly dependent on the environment. Using a surrogate key to reduce the number of columns in the query can provide more uniform performance.

Using a surrogate key can also provide an advantage for tables whose primary key column is based on a character-based data type. Because relational databases are generally more efficient when making numeric comparisons versus character comparisons, by specifying a numeric surrogate key, you can improve query performance.

2.5.3.10. ExtractChangedRecordState SMT

This release introduces the event record changes (**ExtractChangedRecordState**) single message transformation (SMT). You can use this transformation to identify the fields in a Debezium event record whose values changed or remained unchanged after a database operation. To use the transformation, configure it as part of your connector configuration, for example:

```
transforms=changes
transforms.changes.type=io.debezium.transforms.ExtractChangedRecordState
transforms.changes.header.changed=ChangedFields
transforms.changes.header.unchanged=UnchangedFields
```

You can set the following options for this transformation to indicate different types of changes:

header.changed
: Shows the fields changed by an event.

header.unchanged
: Shows the fields that are unchanged by an event.

As in the preceding example, you can set both of these options to separately show both the changed and unchanged fields.

The transformation adds a new header with the specified name, for example, **ChangedFields**. It then sets the header value to a list that contains the names of the changed or unchanged fields.

For more information about using the **ExtractChangedRecordState** SMT, see Event record changes in the Debezium User Guide.

2.5.3.11. Drop event fields with new configuration options for the **ExtractNewRecordState** SMT

You can use the **ExtractNewRecordState** single message transformation (SMT) to convert Debezium change events into a simplified format for consumption by sink connectors.

This release adds three new configuration options for the transformation that you can use to drop fields from the payload or message key of an event:

drop.fields.header.name
: The Kafka message header name to use for listing field names in the source message that are to be dropped.

drop.fields.from.key
: Specifies whether to remove fields also from the key, defaults to **false**.

drop.fields.keep.schema.compatible
: Specifies whether to remove fields that are only optional, defaults to **true**.

NOTE

To maintain schema compatibility in environments that use Avro, the SMT defaults to enforcing schema compatibility. Thus, if you configure a required field to be dropped, the SMT does not remove the field from the key or the payload, unless you disable schema compatibility.

Emitting events that only include changed fields

You can pair the **ExtractChangedRecordState** transformation with the updated **ExtractNewRecordState** SMT to configure a connector to emit events that only include changed fields. The following example shows a configuration that only emits changed columns in an event's payload value:

```
transforms=changes,extract
transforms.changes.type=io.debezium.transforms.ExtractChangedRecordState
transforms.changes.header.unchanged=UnchangedFields
transforms.extract.type=io.debezium.transforms.ExtractNewRecordState
transforms.extract.drop.fields.header.name=UnchangedFields
```

The preceding configuration lists unchanged fields, but ir does not remove them from the event payload. If a field in the specified key did not change, it is retained, because the configuration does not explicitly change the default **false** value for **drop.fields.from.key**.

If the SMT would result in dropping a required field in the event payload, because it did not change, to comply with schema compatibility, the field is retained in the output.

For more information about the **ExtractNewRecordState** SMT, see Extracting source record **after** state from Debezium change events.

2.5.3.12. HeaderToValue SMT

Extracts specified header fields from event records, and then copies or moves the header fields to values in the event record. For more information, see Converting message headers into event record values in the Debezium User Guide.

2.5.3.13. Partition routing SMT

The **PartitionRouting** SMT enables you to route events to specific destination partitions based on the values of one or more specified payload fields. To calculate the destination partition, Debezium generates a hash of the specified field values.

For more information, see Routing records to partitions based on payload fields in the Debezium User Guide.

2.5.4. Technology Preview features

This release introduces the following Technology Preview features:

- Section 2.5.4.1, "MongoDB sharded cluster improvements (Technology Preview)"
- Section 2.5.4.2, "MongoDB incremental snapshots for multi-replica and sharded clusters (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. Technology Preview features provide early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. For more information about support scope, see Technology Preview Features Support Scope.

2.5.4.1. MongoDB sharded cluster improvements (Technology Preview)

In past releases, when you used the Debezium MongoDB connector in a sharded cluster deployment, the connector would open a direct connection with each replica set in a shard. This approach conflict with the MongoDB suggestion that the connector should open a connection with the mongos router instance.

In this release aligns the connector has been redesigned to use the recommended connection strategy. If you use the connector in a sharded cluster, adjust your configuration so that the connector connects to the **mongos** instance. No other changes are required.

2.5.4.2. MongoDB incremental snapshots for multi-replica and sharded clusters (Technology Preview)

You can now use incremental snapshots with MongoDB multi-replica a sharded clusters. For more information, see Incremental snapshots in the MongoDB chapter of the {NameUserGuide}.

Previously available Technology Preview features

The following features that were introduced in earlier releases remain in Technology Preview:

Parallel initial snapshots

You can optionally configure SQL-based connectors to use multiple threads when performing an initial snapshot by setting the **snapshot.max.threads** property to a value greater than 1.

CloudEvents converter

Emits change event records that conform to the CloudEvents specification. The CloudEvents change event envelope can be JSON or Avro and each envelope type supports JSON or Avro as the **data** format.

Custom-developed converters

In cases where the default data type conversions do not meet your needs, you can create custom converters to use with a connector.

Use of the BLOB, CLOB, and NCLOB data types with the Oracle connector

The Oracle connector can consume Oracle large object types.

2.5.5. Developer Preview features

This Debezium 2.3.4 release includes the following Developer Preview features:

- Section 2.5.5.1, "JDBC Sink Connector (Developer Preview)"
- Section 2.5.5.2, "MySQL connector: parallel snapshots (Developer Preview)"
- Section 2.5.5.4, "Exactly-once delivery for PostgreSQL connector (Developer Preview)"
- Section 2.5.5.3, "Oracle connector: Ingesting changes from an Oracle logical standby (Developer Preview)"

IMPORTANT

Developer Preview features are not supported by Red Hat in any way and are not functionally complete or production-ready. Do not use Developer Preview software for production or business-critical workloads. Developer Preview software provides early access to upcoming product software in advance of its possible inclusion in a Red Hat product offering. Customers can use this software to test functionality and provide feedback during the development process. This software might not have any documentation, is subject to change or removal at any time, and has received limited testing. Red Hat might provide ways to submit feedback on Developer Preview software without an associated SLA.

For more information about the support scope of Red Hat Developer Preview software, see Developer Preview Support Scope.

2.5.5.1. JDBC Sink Connector (Developer Preview)

In this release Debezium introduces a JDBC sink connector implementation, breaking with a longstanding focus on developing only source connectors for relational and non-relational databases. The Debezium JDBC sink connector differs from other vendor implementations in that it is capable of ingesting raw change events emitted by Debezium connectors without first applying an event flattening transformation. The Debezium JDBC sink connector can take advantage of native Debezium source connector features, such as column type propagation, enabling you to potentially reduce the processing footprint of your data pipeline, and simplify its configuration.

The following example shows a simple configuration to ingest change events from a Kafka topic called **orders** into a PostgreSQL database. The events in the topic were emitted by a Debezium MySQL connector without using the **ExtractNewRecordState** transformation.

```
{
  "name": "mysql-to-postgres-pipeline",
  "config": {
    "connector_class": "io.debezium.connector.jdbc.JdbcSinkConnector",
    "topics": "orders",
    "connection.url": "jdbc://postgresql://<host>:<port>/<database>",
    "connection.user": "<username>",
    "connection.password": "<password>",
    "insert.mode": "upsert",
    "delete.enabled": "true",
    "primary.key.mode": "record_key",
    "schema.evolution": "basic"
  }
}
```

The preceding example shows a series of **connection.*** properties that define the connection string and credentials for accessing a destination PostgreSQL database. When writing to the destination database, records use *UPSERT* semantics, using an insert to create a record if one doesn’t exist, or updating the record if it does. Schema evolution is enabled and a table’s key columns are derived from the event’s primary key.

You can use this release of the JDBC sink connector with the following relational databases:

- Db2
- MySQL

- Oracle
- PostgreSQL
- SQL Server

For more information, see Debezium connector for JDBC.

2.5.5.2. MySQL connector: parallel snapshots (Developer Preview)

The Debezium initial snapshot process has always been single-threaded for relational databases. This limitation primarily stems from the complexities of ensuring data consistency across multiple transactions.

Beginning in this release, you can configure a connector to use multiple threads when performing a consistent database snapshot. This implementation uses these multiple threads to execute table-level snapshots in parallel.

To take advantage parallel snapshots, set the **snapshot.max.threads** property in the connector configuration, and assign it a value greater than **1**.

Example configuration using parallel snapshots

```
snapshot.max.threads=4
```

Based on the preceding example, the connector snapshots a maximum of 4 tables in parallel. If there are more tables to snapshot, after one thread finishes, the connector processes the next table in the queue. The process continues until all tables have been snapshot.

2.5.5.3. Oracle connector: Ingesting changes from an Oracle logical standby (Developer Preview)

The Debezium connector for Oracle maintains an internal *flush table* to monitor the flush cycles of the Oracle Log Writer Buffer (LGWR) process. The user account through which connector accesses the database must have permission to create and write to the flush table. Logical stand-by databases often have more restrictive rules about data manipulation and may even be read-only, therefore, writing to the database is unfavorable or even not permissible.

To enable the connector to ingest changes from an Oracle read-only logical stand-by database, this release introduces a flag that disables the creation and management of this *flush table*. You can use this Developer Preview feature with both Oracle Standalone and Oracle RAC installations.

To enable the connector to ingest changes from an Oracle read-only logical stand-by, add the following connector option:

```
internal.log.mining.read.only=true
```

2.5.5.4. Exactly-once delivery for PostgreSQL connector (Developer Preview)

Debezium has traditionally been an at-least-once delivery solution, guaranteeing that no change is ever missed. Exactly-once delivery is a proposal by the Apache Kafka community as a part of KIP-618. This proposal aims to address a common problem that producers (source connectors) encounter during a retry. The connector might resend a batch of events to the Kafka broker even though the broker has already committed the batch. This situation can result in duplicate events being sent, which can cause problems for consumers (sink connectors) that are unable to easily handle duplicates.

No connector configuration changes are required to take advantage of exactly-once delivery. However, to enable exactly-once delivery, you must adjust your Kafka Connect worker configuration to use the configuration properties introduced in KIP-618 . In Debezium 2.3.4, exactly-once semantics for PostgreSQL apply only during the streaming phase, not during snapshots.

2.5.6. Other updates in this release

This Debezium 2.3.4 release provides several feature updates and fixes, including the items in the following list:

- DBZ-1973 Enable Debezium to send notifications about its status
- DBZ-2296 Better control of Debezium GTID usage
- DBZ-2979 Connector emits event records after changes to excluded columns
- DBZ-3594 When using **snapshot.collection.include.list**, relational schema isn't populated correctly
- DBZ-4027 Make signalling channel configurable
- DBZ-4488 Failed retriable operations are retried infinitely
- DBZ-4663 Remove option for specifying driver class from MySQL connector
- DBZ-4829 Property **event.processing.failure.handling.mode** is not present in MySQL documentation
- DBZ-5282 Debezium is not working with Apicurio and custom truststores
- DBZ-5283 Add option to exclude unchanged fields in ExtractNewRecordState SMT
- DBZ-5395 Connector offsets do not advance on transaction commit with filtered events when LOB enabled
- DBZ-5490 Document message.key.columns and tombstone events limitations for default REPLICA IDENTITY
- DBZ-5798 Data type conversion failed for MySQL BIGINT
- DBZ-5917 Unable to specify column or table include list if name contains a backslash \
- DBZ-5879 Support retrying database connection failures during connector start
- DBZ-5907 Oracle cannot undo change
- DBZ-5915 PostgreSQL data loss on restarts
- DBZ-5945 Oracle multithreading lost data
- DBZ-5966 Truncate records incompatible with ExtractNewRecordState
- DBZ-5967 Computed partition must not be negative

- DBZ-5973 MongoDB incremental snapshot not working
- DBZ-5985 Table size log message for **snapshot.select.statement.overrides** tables not correct
- DBZ-5988 NPE in execute snapshot signal with **exclude.tables** config on giving wrong table name
- DBZ-5991 There is a problem with PostgreSQL connector parsing the boundary value of money type
- DBZ-5993 Log statement for unparseable DDL statement in MySqlDatabaseSchema contains placeholder
- DBZ-6001 PostgreSQL connector parses the null of the money type into 0
- DBZ-6003 Nullable columns marked with "optional: false" in DDL events
- DBZ-6012 PostgreSQL LSN check should honor **event.processing.failure.handling.mode**
- DBZ-6026 Offsets are not flushed on connect offsets topic when encountering an error on PostgreSQL connector
- DBZ-6029 Unexpected format for TIME column: 8:00
- DBZ-6031 Oracle does not support compression/logging clauses after an LOB storage clause
- DBZ-6037 Debezium is logging the full message along with the error
- DBZ-6039 Improve resilience during internal schema history recovery from Kafka
- DBZ-6046 Add Debezium steps when performing a PostgreSQL database upgrade
- DBZ-6051 Incremental snapshot sends events from the signaling database to Kafka
- DBZ-6064 Mask password in log statement
- DBZ-6075 Loading custom offset storage fails with **Class not found** error
- DBZ-6079 Increase **query.fetch.size** default to something sensible above zero
- DBZ-6084 SQL Server tasks fail if the number of databases is smaller than **maxTasks**
- DBZ-6089 Expose sequence field in CloudEvents message id
- DBZ-6094 Reduce verbosity of skipped transactions if transaction has no events relevant to captured tables
- DBZ-6107 When using LOB support, an UPDATE against multiple rows can lead to inconsistent event data
- DBZ-6112 PostgreSQL: Set Replica Identity when the connector starts
- DBZ-6122 PostgreSQL connector fails when processing toasted varying character arrays and date arrays
- DBZ-6131 Support change stream filtering using MongoDB's aggregation pipeline step

- DBZ-6219 Highlight information about how to configure the schema history topic to store data only for intended tables
- DBZ-6254 Introduce LogMiner query filtering modes
- DBZ-6256 Lock contention on LOG_MINING_FLUSH table when multiple connectors deployed
- DBZ-6329 The **rs_id** field is null in Oracle change event source information block
- DBZ-6353 Using **pg_replication_slot_advance** which is not supported by PostgreSQL10.
- DBZ-6355 **log.mining.transaction.retention.hours** should reference last offset and not **sysdate**
- DBZ-6366 Code Improvements for **skip.messages.without.change**
- DBZ-6379 Toasted **hstore** are not correctly processed
- DBZ-6386 Oracle DDL shrink space for table partition can not be parsed
- DBZ-6396 PostgreSQL connector task fails to resume streaming because replication slot is active
- DBZ-6402 MongoDB connector crashes on invalid resume token
- DBZ-6439 During a snapshot, the Oracle connector takes too long to read structure of captured tables
- DBZ-6457 Oracle parallel snapshots do not properly set PDB context when using multitenancy
- DBZ-6459 [MariaDB] Add support for userstat plugin keywords
- DBZ-6474 Oracle **snapshot.include.collection.list** should be prefixed with **databaseName** in documentation.
- DBZ-6485 Db2 connector can fail with NPE on notification sending
- DBZ-6486 ExtractNewRecordState SMT in combination with HeaderToValue SMT results in Unexpected field name exception
- DBZ-6490 BigDecimal fails when queue memory size limit is in place
- DBZ-6492 Oracle table cannot be captured, got runtime.NoViableAltException
- DBZ-6496 Signal poll interval has incorrect default value
- DBZ-6502 Oracle JDBC driver 23.x throws ORA-18716 - not in any time zone
- DBZ-6509 FileSignalChannel is not loaded
- DBZ-6512 Debezium incremental snapshot chunk size documentation unclear or incorrect
- DBZ-6513 Error value of negative seconds in convertOracleIntervalDaySecond
- DBZ-6515 Debezium incremental snapshot chunk size documentation unclear or incorrect
- DBZ-6524 [PostgreSQL] LTree data is not being captured by streaming

- DBZ-6528 Oracle Connector: Snapshot fails with specific combination
- DBZ-6529 Use better hashing function for PartitionRouting
- DBZ-6533 Table order is incorrect on snapshots
- DBZ-6543 Unhandled NullPointerException in PartitionRouting will crash the whole connect plugin
- DBZ-6559 Bug in **field.name.adjustment.mode** property
- DBZ-6585 Oracle unsupported DDL statement - drop multiple partitions
- DBZ-6589 Support PostgreSQL coercion for UUID, JSON, and JSONB data types
- DBZ-6590 MySQL parser cannot parse **CAST AS dec**
- DBZ-6599 Oracle DDL parser does not properly detect end of statement when comments obfuscate the semicolon
- DBZ-6605 Fixed DataCollections for table scan completion notification
- DBZ-6610 Oracle connector is not recoverable if ORA-01327 is wrapped by another JDBC or Oracle exception
- DBZ-6613 Fatal error when parsing MySQL (Percona 5.7.39-42) procedure
- DBZ-6622 MySQL ALTER USER with RETAIN CURRENT PASSWORD fails with parsing exception
- DBZ-6628 Inaccurate documentation regarding **additional-condition**
- DBZ-6633 Oracle connection SQLRecoverableExceptions are not retried by default
- DBZ-6643 MongoDB connector keeps going up. Fixed via DBZ-6670
- DBZ-6648 Cannot delete non-null interval value
- DBZ-6670 Retriable operations are retried infinitely since error handlers are not reused
- DBZ-6677 Oracle DDL parser does not support column visibility on ALTER TABLE
- DBZ-6690 Should use **topic.prefix** rather than **connector.server.name** in MBean namings
- DBZ-6716 Oracle fails to process a DROP USER
- DBZ-6724 Debezium crashes on parsing MySQL DDL statement (specific JOIN)
- DBZ-6725 ExtractNewDocumentState for MongoDB ignore previous document state when handling delete event's with REWRITE
- DBZ-6733 Oracle LogMiner mining distance calculation should be skipped when upper bounds is not within distance
- DBZ-6736 MariaDB: Unparseable DDL statement (ALTER TABLE IF EXISTS)
- DBZ-6758 When using pgoutput in postgres connector, (+/-)Infinity is not supported in decimal values

- DBZ-6760 Outbox transformation can cause connector to crash
- DBZ-6774 MongoDB New Document State Extraction: nonexistent field for add.headers
- DBZ-6777 Notifications and signals leaks between MBean instances when using JMX channels
- DBZ-6780 Debezium crashes on parsing the MySQL DDL statement (SELECT 1.;)
- DBZ-6794 Debezium crashes on parsing the MySQL DDL statement (SELECT 1 + @sum:=1 AS ss;)
- DBZ-6803 MySQL connector exception because the DDL parser does not accept the REPEAT function
- DBZ-6821 Debezium crashes when DDL statements declare variable names that include non-Latin characters
- DBZ-6824 When parsing MySQL DDL, the connector now properly trims default values for the BIGINT and SMALLINT types
- DBZ-6830 Partial and multi-response transactions are now logged in debug mode only
- DBZ-6867 Streaming aggregation pipeline broken for combination of database filter and signal collection

2.6. DEPRECATED FEATURES

The following features are deprecated in this release:

The **mongodb.hosts** property is no longer supported. To configure Integration connector to connect to a MongoDB replica set, use the **mongodb.connection.string** property.

2.7. KNOWN ISSUES

The following known issue affects Debezium 2.3.4:

- Apicurio registry 2.4.3 and 2.4.4 causes endless rebalance loop on Kafka

CHAPTER 3. SERVICE REGISTRY 2.5 RELEASE NOTES

Service Registry is a data store for standard event schemas and API designs, and is based on the Apicurio Registry open source community project.

NOTE

Red Hat build of Apicurio Registry is now available as part of Red Hat Application Foundations. Red Hat build of Apicurio Registry 2.x and Red Hat Integration Service Registry 2.x are functionally identical. For more information, see Red Hat Application Foundations.

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 create optional rules to govern how Service Registry content evolves over time. These rules include validation of content, integrity of artifact references, and backwards or forwards compatibility of schema or API versions.

3.1. SERVICE REGISTRY INSTALLATION OPTIONS

You can install Service Registry on OpenShift with either of the following data storage options:

- PostgreSQL database
- Red Hat AMQ Streams

For more details, see Installing and deploying Service Registry on OpenShift .

3.2. SERVICE REGISTRY SUPPORTED PLATFORMS

Service Registry 2.5 supports the following core platforms:

- Red Hat OpenShift Container Platform: 4.15, 4.14, 4.13, 4.12
- Red Hat OpenShift Service on AWS: 4.13
- Microsoft Azure Red Hat OpenShift: 4.13
- PostgreSQL: 15, 14, 13, 12
- Red Hat AMQ Streams: 2.6, 2.5, 2.2
- OpenJDK: 17, 11

For more details, see the following article:

- Red Hat Integration Service Registry Supported Configurations .

3.2.1. Supported integration with other products

Service Registry 2.5 also supports integration with the following products:

- Red Hat Single Sign-On (RH-SSO) 7.6

- Red Hat build of Debezium 2.3

3.2.2. Operator metadata versions

For details on the corresponding Service Registry Operator metadata versions used to install and deploy Service Registry, see the following article:

- Red Hat Integration - Service Registry Operator metadata versions .

3.3. SERVICE REGISTRY NEW FEATURES

Service Registry 2.5 includes the following new features:

Service Registry core new features

Upgrade to Quarkus 3.x

- The Service Registry server runtime has been upgraded from Quarkus 2.x to Quarkus 3.x. This upgrade provides improved security, performance, and maintenance. For more details, see https://quarkus.io/quarkus3/. Service Registry 2.5 is built on Quarkus 3.2.

Avro SerDes improvements

- Support for generation of schemas with null fields when using Apache Avro serializers/deserializers. For more details, see Registry-3862.

Schema cache fault tolerance

- Added the option to use an existing schema cache entry instead of throwing an error if schema cache loading fails. For more details, see Registry-3807.

Dereference artifact content

- There are some cases where returning artifact content with referenced content inline might be helpful. For these cases, the Core Registry API v2 adds support for the **dereference** query parameter in certain operations. For more details, see the Apicurio Registry v2 core REST API documentation.

- This support is currently implemented only for Avro and Protobuf artifacts when the **dereference** parameter is specified in the API operation. This parameter is not supported for any other artifact types. For more details, see Registry-2865.

  NOTE

  For Protobuf artifacts, dereferencing content is supported only when all of the schemas belong to the same package.

Service Registry Maven plug-in improvements

- Add the option to skip the **register** goal in the Maven plug-in. For more details, see Registry-3817.

- Automatic detection of references in the Maven plug-in by using the **autoRef** option in the **pom.xml** file. For more details, see Registry-3439. This is a Technology Preview feature.

IMPORTANT

Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

Service Registry Operator new features

Improved support for SQL data source configuration

- Service Registry Operator supports configuring an SQL data source by using environment variables as an alternative to **spec.configuration.sql.dataSource** fields. You can now provide SQL credentials using Kubernetes secrets instead of plaintext in the **ApicurioRegistry** custom resource. For more details, see https://access.redhat.com/solutions/7059053.

- Service Registry Operator has been improved in this version to better support this use case. You can now use both the **spec.configuration.sql.dataSource** and **spec.configuration.env** fields to define parts of the configuration. For example, the following configuration is now valid:

```
apiVersion: registry.apicur.io/v1
kind: ApicurioRegistry
metadata:
  name: myregistry
spec:
  configuration:
    persistence: sql
  sql:
    dataSource:
      url: "jdbc:postgresql://..."
      userName: "postgres-user"
    env:
      - name: REGISTRY_DATASOURCE_PASSWORD
        valueFrom:
          secretKeyRef:
            name: postgres-secret
            key: password
```

  The Operator also detects this type of configuration and applies it immediately without additional user intervention.

Service Registry user documentation and examples

The documentation library has been updated with the new features available in version 2.5:

- Installing and deploying Service Registry on OpenShift
- Migrating Service Registry deployments

- Service Registry User Guide
- Apicurio Registry v2 core REST API documentation

The open source demonstration applications have also been updated:

- https://github.com/Apicurio/apicurio-registry-examples

3.4. SERVICE REGISTRY DEPRECATED FEATURES

Service Registry core deprecated features

- *Confluent Schema Registry API version 6 (compatibility API)* : Service Registry currently supports two versions of the Confluent Schema Registry API on separate endpoints: version 6 and version 7. The v6 API endpoint is deprecated, and will be removed in a future release. Ensure that you replace all references to the v6 API endpoint with references to the v7 API endpoint.
- *Service Registry Core API version 1* : Service Registry support for the original version 1 of the Service Registry Core API is now deprecated. This v1 legacy API will be removed in the next major release.
- *Dynamic log level configuration*: The **/admin/loggers** and **/admin/loggers/{logger}** API endpoints are now deprecated in the v2 Service Registry Core API. These endpoints will be removed in a future release.
- *Registry V1 export utility* : Service Registry support for the command-line export utility is now deprecated. The export tool, which is used to export data from Service Registry 1.x into a format that can be imported into 2.x, will no longer be released or maintained. All customers should have already upgraded from 1.x to 2.x.

Service Registry Operator deprecated features

- *JAVA_OPTIONS environment variable*: The **JAVA_OPTIONS** environment variable is no longer the preferred way to configure Java options for Service Registry. You can use the **JAVA_OPTS_APPEND** environment variable instead. The **JAVA_OPTS** environment variable is also available, which replaces the default content of Java options. However, it is best to avoid using **JAVA_OPTS** because it might interfere with some Service Registry Operator functionality.
- *Setting environment variables by editing the Deployment resource* : In previous versions, you could set environment variables for Service Registry by directly editing its **Deployment** resource, which was supported by the Service Registry Operator. Now that you can manage environment variables by using the **spec.configuration.env** field in the **ApicurioRegistry** CRD file, the previous procedure is deprecated and the Operator support for it will be removed. Ensure that you use the **spec.configuration.env** field to set all environment variables that are not set by the Operator.
- *Retention of environment variables for features that are not enabled* : The Service Registry Operator sets environment variables to enable and configure various features, such as Salted Challenge Response Authentication Mechanism (SCRAM) security when using Kafka storage. When such features are disabled, the Operator currently retains the associated environment variables, which can cause problems. Retention of such environment variables is deprecated, and the Operator support for it will be removed. Ensure that your deployment does not rely on the retention of such environment variables.
- *Environment variable precedence* : The Service Registry Operator might attempt to set an environment variable that is already explicitly specified in the **spec.configuration.env** field. If an

environment variable has a conflicting value, the value set by the Service Registry Operator takes precedence by default. This behavior will change in the future, to enable users to overwrite most environment variables set by the Operator. Ensure that your deployment does not rely on the original precedence behavior.

3.5. UPGRADING AND MIGRATING SERVICE REGISTRY DEPLOYMENTS

You can upgrade the Service Registry server automatically from Service Registry 2.x to Service Registry 2.5 on OpenShift. There is no automatic upgrade from Service Registry 1.x to Service Registry 2.x, and a migration process is required.

3.5.1. Updating 2.x client dependencies

It is not mandatory to update client dependencies for this release. Existing Service Registry 2.x client applications continue to work with Service Registry 2.5.

However, before the next release of Service Registry, you must update all of your client dependencies to use the latest version of Service Registry. Client dependencies include dependencies for the Service Registry Kafka serializers/deserializers (SerDes), Maven plug-in, and Java client applications.

For example, to update the Maven dependencies for a Java client application, specify the version in your **pom.xml** file as follows:

```
<dependency>
    <groupId>io.apicurio</groupId>
    <artifactId>apicurio-registry-client</artifactId>
    <version>2.5.10.Final-redhat-00001</version>
</dependency>
```

For more details, see Legacy REST API date formats enabled by default .

3.5.2. Upgrading from Service Registry 2.x on OpenShift

You can upgrade from Service Registry 2.x on OpenShift 4.11 to Service Registry 2.5 on OpenShift 4.12 or later. You must upgrade both your Service Registry and your OpenShift versions, and upgrade OpenShift one minor version at a time.

Prerequisites

- You already have Service Registry 2.x installed on OpenShift 4.11 or later.
- You have backed up your existing Service Registry storage data in your Kafka topic or PostgreSQL database. For more details, see Installing and deploying Service Registry on OpenShift.

IMPORTANT

In production environments on OpenShift, to help ensure that storage is backed up before upgrading, it is best to set the Operator update approval strategy for Service Registry to manual instead of automatic.

Procedure

1. In the OpenShift Container Platform web console, click **Administration** and then **Cluster Settings**.
2. Click the pencil icon next to the **Channel** field, and select the next minor **candidate** version (for example, change from **stable-4.11** to **candidate-4.12**).
3. Click **Save** and then **Update**, and wait until the upgrade is complete.
4. If the OpenShift version is less than 4.13, repeat steps 2 and 3, and select **candidate-4.13** or later.
5. Click **Operators** > **Installed Operators** > **Red Hat Integration - Service Registry**.
6. Ensure that the **Update channel** is set to **2.x**.
7. If the **Update approval** is set to **Automatic**, the upgrade should be approved and installed immediately after the **2.x** channel is set.
8. If the **Update approval** is set to **Manual**, click **Install**.
9. Wait until the Operator is deployed and the Service Registry pod is deployed.
10. Verify that your Service Registry system is up and running.

Additional resources

- For more details on how to set the Operator update channel in the OpenShift Container Platform web console, see Changing the update channel for an Operator.

3.5.3. Migrating from Service Registry 1.1 on OpenShift

For details on migrating from Service Registry 1.1 to Service Registry 2.x, see Migrating Service Registry deployments.

3.6. SERVICE REGISTRY RESOLVED ISSUES

Table 3.1. Resolved issues in Service Registry 2.5.10

Issue	Description
IPT-1091	Service Registry Operator should support both the JAVA_OPTS_APPEND and JAVA_OPTIONS (deprecated) environment variables.
IPT-1092	Service Registry upgrade breaks Confluent v6 compatibility API.

Table 3.2. Resolved issues in Service Registry 2.5.9

Issue	Description
IPT-1071	Possible data loss when upgrading Service Registry with KafkaSQL storage and Protobuf artifacts with references.

Issue	Description
IPT-1035	Service Registry Pods in CrashLoop after Operator upgrade to 2.2.3.
Registry-4417	Orphaned content not properly deleted from Service Registry.
Registry-4283	Service Registry server should fail when two references with the same name are created for a single artifact.
Registry-4226	Delete all rules REST API operation doesn't delete the **INTEGRITY** rule (KafkaSQL storage only).
Registry-4215	Avro with different field order is considered equal in canonical form.
Registry-4107	Validity, compatibility, and integrity rule values are not displayed in the Apicurio Registry web console in read-only mode.

Table 3.3. Resolved issues in Service Registry 2.5.5

Issue	Description
Registry-4104	When AMQ Streams storage and OAuth are configured, Service Registry fails to start due to missing kafka-oauth-client class.

Table 3.4. Resolved issues in Service Registry 2.5.4

Issue	Description
Registry-4019	Some health checks are always UP even when a counter hits the limit.
Registry-3956	Schema registry is called even when the schema already exists in the local cache (SerDes).
Registry-3725	Resource owner password grant - basic auth - java.lang.IllegalStateException: Client is closed.
Registry-3647	Protobuf content canonicalHash outdated value detected.

3.7. SERVICE REGISTRY RESOLVED CVES

The following Common Vulnerabilities and Exposures (CVEs) are resolved in Service Registry 2.5:

Table 3.5. CVEs resolved in Apicurio Registry 2.5.9

CVE	Description
CVE-2024-20952 CVE-2024-20921 CVE-2024-20919 CVE-2024-20918	Difficult to exploit vulnerability allows unauthenticated attacker with network access via multiple protocols to compromise Oracle Java SE, Oracle GraalVM for JDK, Oracle GraalVM Enterprise Edition.
CVE-2024-20945	Difficult to exploit vulnerability allows low privileged attacker with logon to the infrastructure where Oracle Java SE, Oracle GraalVM for JDK, Oracle GraalVM Enterprise Edition executes to compromise Oracle Java SE, Oracle GraalVM for JDK, Oracle GraalVM Enterprise Edition.
CVE-2024-20932	Easily exploitable vulnerability allows unauthenticated attacker with network access via multiple protocols to compromise Oracle Java SE, Oracle GraalVM for JDK, Oracle GraalVM Enterprise Edition.
CVE-2023-39615	A flaw was found in Libxml2, where it contains a global buffer overflow via the xmlSAX2StartElement() function at /libxml2/SAX2.c.
CVE-2023-38473	A vulnerability was found in Avahi. A reachable assertion exists in the avahi_alternative_host_name() function.
CVE-2023-38472	A vulnerability was found in Avahi. A reachable assertion exists in the avahi_rdata_parse() function.
CVE-2023-38471	A vulnerability was found in Avahi. A reachable assertion exists in the dbus_set_host_name function.
CVE-2023-38470	A vulnerability was found in Avahi. A reachable assertion exists in the avahi_escape_label() function.
CVE-2023-38469	A vulnerability was found in Avahi, where a reachable assertion exists in avahi_dns_packet_append_record.
CVE-2023-27043	The email module of Python through 3.11.3 incorrectly parses e-mail addresses that contain a special character.
CVE-2023-7104	A vulnerability was found in SQLite3. This issue affects the sessionReadRecord function of the ext/session/sqlite3session.c function in the make alltest Handler component. Manipulation may cause a heap-based buffer overflow to occur.
CVE-2023-5981	A vulnerability was found that the response times to malformed ciphertexts in RSA-PSK ClientKeyExchange differ from response times of ciphertexts with correct PKCS#1 v1.5 padding.
CVE-2023-5678	A flaw was found in OpenSSL, which caused the generation or checking of long X9.42 DH keys or parameters to be much slower than expected. This issue could lead to a denial of service.

CVE	Description
CVE-2023-5388	It was discovered that the numerical library used in NSS for RSA cryptography leaks information whether high order bits of the RSA decryption result are zero.
CVE-2023-3817 CVE-2023-3446	A vulnerability was found in OpenSSL. This security issue occurs because the applications that use the DH_check(), DH_check_ex(), or EVP_PKEY_param_check() functions to check a DH key or DH parameters may experience long delays.
CVE-2022-48564	A vulnerability was found in the Python core plistlib library within the read_ints() function in the plistlib.py file.
CVE-2022-48560	A use-after-free vulnerability was found in Python via the heappushpop function in the heapq module.
CVE-2021-3468	A flaw was found in avahi. The event used to signal the termination of the client connection on the avahi Unix socket is not correctly handled in the client_work function, allowing a local attacker to trigger an infinite loop.

Table 3.6. Resolved CVE issues in Service Registry 2.5.4

Issue	Description
IPT-1034	CVE-2023-5072 JSON-java: parser confusion leads to OOM error.
IPT-1030	CVE-2023-31582 jose4j: Insecure iteration count setting.
IPT-1021	CVE-2023-44487 undertow: HTTP/2: Multiple HTTP/2 enabled web servers are vulnerable to a DDoS attack (Rapid Reset Attack).
IPT-1013	CVE-2023-39410 avro: apache-avro: Apache Avro Java SDK: Memory when deserializing untrusted data in Avro Java SDK.
IPT-995	CVE-2023-4853 quarkus-vertx-http: quarkus: HTTP security policy bypass.
IPT-993	CVE-2023-39321 CVE-2023-39322 integration-service-registry-operator-container: various flaws.
IPT-953	CVE-2023-29409 integration-service-registry-operator-container: golang: crypto/tls: slow verification of certificate chains containing large RSA keys.
IPT-948	CVE-2023-29406 integration-service-registry-operator-container: golang: net/http: insufficient sanitization of Host header.
IPT-940	CVE-2023-34462 netty: SniHandler 16MB allocation leads to OutOfMemoryError.
IPT-936	CVE-2023-34455 snappy-java: Unchecked chunk length leads to DoS.

Issue	Description
IPT-935	CVE-2023-35116 jackson-databind: denial of service via cyclic dependencies.
IPT-874	CVE-2023-1584 quarkus-oidc: ID and access tokens leak via the authorization code flow.

Table 3.7. Additional CVEs resolved in Apicurio Registry 2.5.4

CVE	Description
CVE-2023-44483	All versions of Apache Santuario - XML Security for Java prior to 2.2.6, 2.3.4, and 3.0.3, when using the JSR 105 API, are vulnerable to an issue where a private key may be disclosed in log files when generating an XML Signature and logging with debug level is enabled.
CVE-2023-43642	A flaw was found in SnappyInputStream in snappy-java, a data compression library in Java. This issue occurs when decompressing data with a too-large chunk size due to a missing upper bound check on chunk length.
CVE-2023-42503	Apache Commons Compress: Denial of service via CPU consumption for malformed TAR file.
CVE-2023-40217	Python 3 ssl.SSLSocket is vulnerable to a bypass of the TLS handshake in certain instances for HTTPS servers and other server-side protocols that use TLS client authentication such as mTLS.
CVE-2021-39194	Denial of service while parsing polymorphic input with tagged polymorphism style in kaml
CVE-2023-34454 CVE-2023-34453	A flaw was found in Snappy-java's shuffle function, which does not check input sizes before beginning operations.
CVE-2023-29491	A vulnerability was found in ncurses and occurs when used by a setuid application.
CVE-2023-28118	kaml has potential denial of service while parsing input with anchors and aliases.
CVE-2022-24823	When using multipart decoders in netty, local information disclosure can occur via the local system temporary directory if temporary storing of uploads on the disk is enabled.
CVE-2023-4911	A buffer overflow was discovered in the GNU C Library's dynamic loader ld.so while processing the GLIBC_TUNABLES environment variable.
CVE-2023-4813	A flaw was found in glibc. In an uncommon situation, the gaih_inet function may use memory that has been freed, resulting in an application crash.
CVE-2023-4806	A flaw was found in glibc. In an extremely rare situation, the getaddrinfo function may access memory that has been freed, resulting in an application crash.

CVE	Description
CVE-2023-4527	A flaw was found in glibc. When the getaddrinfo function is called with the AF_UNSPEC address family and the system is configured with no-aaaa mode via /etc/resolv.conf, a DNS response via TCP larger than 2048 bytes can potentially disclose stack contents through the function returned address data, and may cause a crash.

3.8. SERVICE REGISTRY KNOWN ISSUES

The following known issues apply in Service Registry 2.5:

Service Registry core known issues

Registry-3413 - Legacy REST API date formats enabled by default

For maximum compatibility and for easier upgrades from older versions of Service Registry, the date format used in the Service Registry REST API is not compliant with OpenAPI standards. This is because of a bug in older versions.

Before the next release of Service Registry, you must upgrade all of your client applications to use the latest Service Registry client version. The next release will fix the date format bug, which will result in older clients no longer being compatible with the REST API.

To update your REST API to be OpenAPI compliant, you can fix the date format bug in *this* version of Service Registry as follows:

1. Update all of your client applications to version **2.5.10.Final-redhat-00001**, as described in Updating 2.x client dependencies.
2. Set the following environment variable to the value shown:

```
REGISTRY_APIS_V2_DATE_FORMAT=yyyy-MM-dd'T'HH:mm:ss'Z'
```

IPT-814 - Service Registry logout feature incompatible with RH-SSO 7.6

In RH-SSO 7.6, the **redirect_uri** parameter used with the logout endpoint is deprecated. For more details, see the RH-SSO 7.6 Upgrading Guide . Because of this deprecation, when Service Registry is secured by using the RH-SSO Operator, clicking the **Logout** button displays the **Invalid parameter: redirect_uri** error.

For a workaround, see https://access.redhat.com/solutions/6980926.

IPT-701 - CVE-2022-23221 H2 allows loading custom classes from remote servers through JNDI

When Service Registry data is stored in AMQ Streams, the H2 database console allows remote attackers to execute arbitrary code by using the JDBC URL. Service Registry is not vulnerable by default and a malicious configuration change is required.

Service Registry Operator known issues

Operator-42 - Autogeneration of OpenShift route might use wrong base host value

If multiple **routerCanonicalHostname** values are specified, autogeneration of the Service Registry OpenShift route might use a wrong base host value.

APPENDIX A. USING YOUR SUBSCRIPTION

Integration is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

ACCESSING YOUR ACCOUNT

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

ACTIVATING A SUBSCRIPTION

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

DOWNLOADING ZIP AND TAR FILES

To access zip or tar files, use the customer portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Scroll down to **INTEGRATION AND AUTOMATION**.
3. Click **Red Hat Integration** to display the Red Hat Integration downloads page.
4. Click the **Download** link for your component.

Revised on 2024-03-22 13:11:14 UTC