Red Hat build of Debezium 2.1.4

Release Notes for Red Hat build of Debezium 2.1.4

What's new in Red Hat build of Debezium

Last Updated: 2023-04-21

Red Hat build of Debezium 2.1.4 Release Notes for Red Hat build of Debezium 2.1.4

What's new in Red Hat build of Debezium

Legal Notice

Abstract

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

Table of Contents

CHAPTER 1. RED HAT BUILD OF DEBEZIUM

Red Hat build of Debezium 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 build of Debezium 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 build of Debezium 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 2.1.4 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 2.1.4”
- Section 2.5, “New Debezium features”

2.1. DEBEZIUM DATABASE CONNECTORS

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

- Db2
- 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 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.
- MongoDB
  - Currently, you cannot use the transaction metadata feature of the Debezium MongoDB connector with MongoDB 4.2.
- Oracle

    - The Debezium Oracle connector does not include the Oracle JDBC driver (**ojdbc8.jar**). See the 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 2.1.4 Supported configurations page .

2.2.1. AMQ Streams API version

Debezium runs on AMQ Streams 2.3.

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 2.1.4

The current version of Debezium includes changes that require you to follow specific steps when you upgrade from an earlier version. For more information, refer to the list of breaking changes and the upgrade procedure.

2.4.1. Upgrading connectors to Debezium 2.1.4

Debezium 2.1.4 is the first Red Hat release of a new Debezium major release version. Some of the changes in the Debezium 2.1.4 are not backward-compatible with previous versions of Debezium. As a result, to preserve data and ensure continued operation when you upgrade from Debezium 1.x versions to 2.1.4, 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, 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 3.3.1. This is the default Kafka version in AMQ Streams 2.3.
- The Java 11 runtime is required and must be available prior to upgrading. AMQ Streams 2.3 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 2.3.
- Check the backward-incompatible changes in the Breaking changes list.
- Verify that your environment complies with the Debezium 2.1.4 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 2.1.4. Refer to Table 2.1, “Updates to connector configuration properties” 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 2.1.4.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. See Changes to Debezium storage in the list of Breaking changes.
7. Restart Kafka Connect to start the connector. After you restart the connector, the 2.1.4.Final connector continues to process events from the point at which you stopped the connector before the upgrade. Change events records that the connector wrote to Kafka before the upgrade are not modified.

2.5. NEW DEBEZIUM FEATURES

Debezium 2.1.4 includes the following updates.

- Section 2.5.1, “Breaking changes”
- Section 2.5.2, “Features promoted to General Availability”

2.5.1. Breaking changes

The following changes in Debezium 2.1.4 represent significant differences in connector behavior and require configuration changes that are not compatible with earlier Debezium versions:

Changes that apply to multiple connectors

Database history topic

Now referred to as the *database schema history* topic.

Limits on object sizes for memory queues

Sizes are no longer calculated by using reflection. Instead, queue limits are estimated based on the message schema. (DBZ-2766) (MongodB, MySQL, Oracle, PostgreSQL, SQL Server)

Exposure of connector metrics

Debezium previously exposed connector metrics as a single tuple of snapshot, streaming, and history-based beans. With this release, connector metrics are now exposed as a multi-partition scheme. As a result, metrics names, and the way in which they are exposed is changed (DBZ-4726). If you use Grafana, Prometheus, or similar JMX frameworks for gathering metrics, review your process for collecting metrics.

database.server.name property

No longer used in the connector configuration. For more information, see Table 2.1, “Updates to connector configuration properties”.

Schema definition

For naming and versioning consistency, Debezium schemas are now defined in a central point (DBZ-4365, DBZ-5044). If you use a schema registry, schema compatibility issues might occur.

Debezium storage changes

In previous releases, Debezium supported reading and storing offsets, history, and other data as a part of the debezium-core module. This release includes a new **debezium-storage** module with implementations for storing data in a local file system or in Kafka (DBZ-5229). The extension point implemented in this approach makes it possible to introduce other storage implementations in the future. As part of the upgrade, you might need to adjust your application’s dependencies depending on the storage module implementations required by the code.

Restart after communication exceptions

After an exception related to communication (SqlException, IOException) is thrown, by default, the Debezium MongoDB, MySQL, PostgreSQL, and SQL Server connectors now restart automatically (DBZ-5244).

Default value of theskipped.operations configuration option

The default value is now **truncate** (DBZ-5497) (MongoDB, MySQL, Oracle, PostgreSQL, SQL Server)

Default value of schema.name.adjustment.mode property

The default value is now **none** (DBZ-5541). The previous default option, **avro** was a good choice for customers who use the Avro converter, but it caused confusion in environments that use the JSON converter. As part of this change, the **sanitize.field.names** property is no longer available.

*** Removal of connector configuration properties**

Several properties that were available in Debezium 1.x versions are no longer valid and have been replaced by new properties. For more information, see the following table:

Table 2.1. Updates to connector configuration properties

1.x property	Equivalent 2.x property
database.* (pass-through database driver properties) (DBZ-5043)	**driver.***
database.dbname (SQL Server)	**database.names**
database.history.consumer.* (DBZ-5043)	**schema.history.internal.consumer.***
database.history.kafka.bootstrap.servers (DBZ-5043)	**schema.history.internal.kafka.bootstrap.servers**
database.history.kafka.topic (DBZ-5043)	**schema.history.internal.kafka.topic**
database.history.producer.* (DBZ-5043)	**schema.history.internal.producer.***
database.server.name (DBZ-5043)	**topic.prefix**
mongodb.name (MongoDB)	**topic.prefix**
schema_blacklist (DBZ-5045)	**schema_exclude_list**
schema_whitelist (DBZ-5045)	**schema_include_list**

Changes that apply to the MySQL connector

- The MySQL connector no longer supports legacy JDBC legacy date/time properties (DBZ-4965).

Changes that apply to the MongoDB connector

- The MongoDB connector no longer supports streaming directly from the oplog. Change streams represents a superior mechanism for performing change data capture with MongoDB. Rather than reading the oplog directly, the connector now delegates the task of capturing and decoding the oplog data to MongoDB change streams, which expose the changes that occur within a collection as an event stream. The Debezium connector subscribes to the stream and then delivers the changes downstream to Kafka. The transition to change streams offers a variety of benefits, including the ability to stream changes from non-primary nodes, and the ability to emit update events with a full document representation for downstream consumers.
- The configuration property **mongodb.name** is replaced by the **topic.prefix** property.

Changes that apply to the PostgreSQL connector

- Protocol buffer (**protobuf**) decoding is no longer supported (DBZ-703).
- The **wal2json** plugin is no longer supported (DBZ-4156).

- The PostgreSQL transaction id is now 32-bit integer that rolls over. To simplify de-duplication of transactions, the LSN is now included as part of the identifier (DBZ-5329).

Changes that apply to the SQL Server connector

- If SSL is not enabled for a SQL Server database, or if you want to connect to the database without using SSL, disable SSL by setting the value of the **database.encrypt** property in the connector configuration to **false**.
- The **database.dbname** property is replaced by the **database.names** property.

2.5.2. Features promoted to General Availability

The following features are promoted from Technology Preview to General Availability in the Debezium 2.1.4 release:

- MongoDB **ExtractNewDocumentState** SMT

2.5.3. Debezium feature updates

This Debezium 2.1.4 release provides several feature updates and fixes, including the items in the following list:

- The MySQL connector now supports binlog compression. DBZ-2663
- Limit log output for "Streaming requested from LSN" warnings. DBZ-3007
- Implements support for JSON_TABLE in the MySQL connector parser. DBZ-3575
- You can now pause or stop incremental snapshots by sending a signal. DBZ-4251
- The SQL Server connector now fails when the user account lacks the required CDCReader permission. DBZ-4346
- The MongoDB connector can now decode binary payloads DBZ-4600
- You can now pause and resume a running incremental snapshot DBZ-4727
- You can now specify the MongoDB connection settings by specifying a connection string URI DBZ-4733
- The **field.exclude.list** property for the MongoDB connector now works with fields from different collections that have the same name. DBZ-4846
- The PostgresSQL connector now retries the connection after the error **PSQLException: This connection has been closed.** DBZ-4948
- The MySQL connector now stores the event header timestamp in the the history record DBZ-4998
- The LogMiner batch size is now adjusted based on the current batch size, rather than the default size. DBZ-5005
- You can now configure the maximum number of entries to cache for the ByLogicalTableRouter SMT. DBZ-5072

- A new extension API permits you to query the Debezium version. DBZ-5092
- Adds the field **ts_ms** to schema change events to identify when an event occurs or is processed. DBZ-5098
- When the MongoDB connector converts oplog entries, it now uses the **RawBsonDocument** class rather than **Document**. DBZ-5113
- MySQL commit timestamp DBZ-5170
- The event SCN is now included in Oracle event records. DBZ-5225
- To avoid occurrences of **UnknownTopicOrPartitionException**, you can now set **database.history.kafka.create.timeout.ms** to specify how long the connector waits for the Kafka history topic to be created. DBZ-5249
- After modifying the primary key, LOB type data is now consistent between the source and sink. DBZ-5295
- The MySQL connector now retries after it receives an error when attempting to read the binlog. DBZ-5333
- During an incremental snapshot, the Oracle connector now correctly parses events from a database with a name that includes a period. DBZ-5336
- Support PostgreSQL default value function calls with schema prefixes. DBZ-5340
- The MySQL connector fails to convert unsigned **tinyint** data types for MySQL 8.x. DBZ-5343
- The Oracle connector logs a warning when it detects an unsupported LogMiner operation for a captured table. DBZ-5351
- The Oracle connector throws a NullPointerException when a unique index is based on both system and non-system generated columns. DBZ-5356
- Fixes a problem in which column hash v2 did not work with the MySQL connector. DBZ-5366
- Fixes a problem in which JSON expansion failed for outbox event payloads that contain nested arrays in which the first array contain no elements. DBZ-5367
- Fixes MongoDB connector connection failures when using AWS DocumentDB with MongoDB compatibility. DBZ-5371
- Fixes a problem in which the Oracle connector logged CommitScn in an unexpected format. DBZ-5381
- Fixes the PostgreSQL connector error org.postgresql.util.PSQLException: Bad value for type timestamp/date/time: CURRENT_TIMESTAMP. DBZ-5384
- Fixes a problem with the MySQL connector in which the **previousID** property is missing in the history topic. DBZ-5386
- Check constraint introduces a column based on constraint in the schema change event. DBZ-5390
- Fixes a problem that occurs when the PostgreSQL connector captures a column that is referenced as the PRIMARY KEY, but no matching column is defined in table. DBZ-5398

- Clarifies documentation for **signal.data.collection** when using Oracle with pluggable database support DBZ-5399
- The PostgreSQL connector now uses GMT to specify timestamps. DBZ-5403
- Ad hoc and incremental snapshots now support an **additional-condition** parameter for specifying subsets of data to capture. DBZ-5327
- Adds logic to enable the Oracle connector to gracefully unsupported non-relational tables during streaming DBZ-5441
- The SQL Server connector task now restarts after a "Socket closed" exception. DBZ-5478
- Augment a uniqueness key field/value in regular expression topic naming strategy. DBZ-5480
- MySqlErrorHandler should handle SocketException DBZ-5486
- The MySQL connector now adds database column comments to the connector schema. DBZ-5489
- Expose default values and enum values in schema history messages. DBZ-5511
- Support BASE64_URL_SAFE in BinaryHandlingMode DBZ-5544
- Supply partition when committing offsets with source database DBZ-5557
- Traditional snapshot process setting **source.ts_ms**. DBZ-5591
- Clean up "logical name" configuration. DBZ-5594
- The MySQL Connector now captures TRUNCATE events. DBZ-5610
- Clarify semantics of include/exclude options in documentation. DBZ-5625
- You can now configure the MongoDB connector to include a **before** field when it emits change events. DBZ-5628
- Logging enhancement for non-incremental snapshot in PostgreSQL connector. DBZ-5639
- Improve LogMiner query performance by reducing REGEXP_LIKE disjunctions. DBZ-5648
- You can configure how often the MongoDB connector attempts to send heartbeat messages to the server. DBZ-5736
- Enhance the ability to sanitize topic name DBZ-5790
- You can now configure **flush.lsn.source** to prevent the PostgreSQL connector from automatically committing the LSN of processed records to the database. DBZ-5811
- You can now use the **ComputePartition** SMT to route data to specific topic partitions based on certain fields. DBZ-5847
- You can now configure the **event.processing.failure.handling.mode** to enable the PostgreSQL connector to skip failed LSN checks. DBZ-6012
- Connector offsets now advance correctly in when you use the Oracle connector in pluggable database deployments (CDB) that have infrequent changes. DBZ-6125

2.6. TECHNOLOGY PREVIEW FEATURES

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.

Debezium includes the following Technology Preview features:

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.

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.

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. The CloudEvents change event envelope supports Avro encoding change event envelope can be JSON or Avro and each envelope type supports JSON or Avro as the **data** format.

Content-based routing

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

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.

Filter SMT

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

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.

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

The Oracle connector can consume Oracle large object types.