OpenCDC record

An OpenCDC record in Conduit aims to standardize the format of data records exchanged between different connectors within a data processing pipeline. The primary objective is to ensure compatibility between various combinations of source and destination connectors.

Benefits

1. Support for Operations
2. : The format should support representing records forcreate
3. ,update
4. ,delete
5. , andsnapshot
6. operations.
7. Standard Metadata Fields
8. : Definition of standard metadata fields to provide essential information about each record. These can vary depending on the record. SeeMetadata Fields
9. for more information.
10. Integration with Data Tools
11. : We believe that being strict about the record format Conduit consumes and produces will make it easier to integrate with other data processing tools.

Fields

- .Position
- uniquely represents the position of record. This is used to track the position of a record in a source connector, enabling Conduit to resume a stopped pipeline.
- .Operation
- defines what triggered the creation of a record. There are four possibilities:create
- ,update
- ,delete
- orsnapshot
- . The first three operations are encountered during normal CDC operation, whilesnapshot
- is meant to represent records during an initial load. Depending on the operation, the record will contain either the payload before the change, after the change, both or none (see fields.Payload.Before
- and.Payload.After
-).
- .Key
- represents a value that should identify the entity (e.g. database row).
- .Metadata
- contains additional information regarding the record.
- .Payload.Before
- holds the payload before the operation ocurred. These could be present in operations such asupdate
- anddelete
- .
- .Payload.After
- holds the payload after the operation ocurred. These could be present in operations such ascreate
- ,snapshot
- orupdate
- .

note We're indicating.Position , and not.position as defined in itsRecord message , to show itsGo template notation as used by theGo representation of an OpenCDC record. This field is public and must start with an uppercase letter.

Representation

Conduit relies onProtocol Buffers (protobuf) when it comes to defining an OpenCDC record to benefit from the several advantages that it provides. Its definition can be found in theBuf Schema Registry .

When processing records in Conduit, you can always expect a similar structure to the following:

{ "position" :

"c3RhbmRpbmc=" , "operation" :

"update" , "metadata" :

{ "file.path" :

"./example.in" , "opencdc.readAt" :

"1663858188836816000" , "opencdc.version" :

"v1" } , "key" :

"cGFkbG9jay1rZXk=" , "payload" :

{ "before" :

"eWVsbG93" , "after" :

{ "bool" :

true , "float32" :

1.2 , "float64" :

1.2 , "int" :

1 , "int32" :

1 , "int64" :

1 , "string" :

"orange" } } } note .Position ,.Key , and.Payload.Before are represented asBase64 encoded in the example above because these will be a byte slice when represented as JSON.

Metadata fields

As part of an OpenCDC record, there will be a set of fields provided that will vary depending on the connector. These fields can be common to allOpenCDC records as part of our standard, some related toConduit , and others that will be provided by eachConnector implementation independently. These fields can be useful to define conventions that will be then used by Conduit to expand its functionality. Notice that all these fields use a dot notation syntax to indicate what they refer to, preventing accidental clashes. Here are the ones you can find:

opencdc.createdAt

Contains the time when the record was created in the 3rd party system. The expected format is a Unix timestamp in nanoseconds.

{ // other record fields "metadata": { "opencdc.createdAt": "1663858188836816000", // rest of metadata }, // other record fields }

opencdc.readAt

Contains the time when the record was read from the 3rd party system. The expected format is a Unix timestamp in nanoseconds.

{ // other record fields "metadata": { "opencdc.readAt": "1663858188836816000", // rest of metadata }, // other record fields }

opencdc.version

Contains the version of the OpenCDC format (e.g., "v1"). This field exists to ensure the OpenCDC format version can be easily identified in case the record gets marshaled into a different untyped format (e.g. JSON).

{ // other record fields "metadata": { "opencdc.version": "v1", // rest of metadata }, // other record fields }

opencdc.collection

Contains the name of the collection from which the record originated and/or where it should be written to.

note It's up to the connector to populate this field. In other words, not all records may have this field. { // other record fields "metadata": { "opencdc.collection": "employees", // rest of metadata }, // other record fields }

conduit.source.plugin.name

The name of the source plugin that created the record.

{ // other record fields "metadata": { "conduit.source.plugin.name": "builtin:file", // rest of metadata }, // other record fields }

conduit.source.plugin.version

The version of the source plugin that created the record.

{ // other record fields "metadata": { "conduit.source.plugin.version": "v1.0.2", // rest of metadata }, // other record fields }

conduit.source.connector.id

conduit.source.connector.id is the ID of the source connector that received the record.

{ // other record fields "metadata": { "conduit.source.connector.id": "connectorID", // rest of metadata }, // other record fields }

conduit.destination.plugin.name

The name of the destination plugin that has written the record.

{ // other record fields "metadata": { "conduit.destination.plugin.name": "builtin:file", // rest of metadata }, // other record fields }

conduit.destination.plugin.version

The version of the destination plugin that has written the record.

{ // other record fields "metadata": { "conduit.destination.plugin.version": "v0.9.1", // rest of metadata }, // other record fields }

conduit.dlq.nack.error

Contains the error that caused a record to be nacked and pushed to thedead-letter queue (DLQ) .

conduit.dlq.nack.node.id

The ID of the internal node that nacked the record.

Connector-specific metadata

These metadata fields will be provided by each connector implementation allowing them to add any necessary metadata. As previously mentioned, to avoid unintended conflicts of metadata keys, the convention these will follow are the same as before, indicating first the connector name that's adding them.

Taking the sameprevious record example , you'll notice there is a metadata key namedfile.path , which would indicate this field was added by afile plugin.

{ // other record fields "metadata": { "file.path": "./example.in", // rest of metadata }, // other record fields }Edit this page
Previous Metrics Next Pipeline Semantics