Output Formats

One of the challenges to be solved when integrating Conduit with other systems, such as Kafka Connect and Debezium, is the data format. This is present in situations where raw data needs to be written, for example when writing messages to Kafka. By default, Conduit uses theOpenCDC format . Conduit also makes it possible to change the output format so that the data can be consumed by other systems.

note Configuring different output formats is possible for connectors which use Conduit SDK's default middleware. Please consult the connector's documentation to confirm the feature is available. The output format is configured at the level of a destination connector. This implies that different connectors in the same pipeline can use different formats.

Configuration parameters

There are two connector configuration parameters which control the output format:

- sdk.record.format
- : used to choose the format itself
- sdk.record.format.options
- : used to configure the specifics of the chosen format.

Available formats

Most formats come in the form ofname/encoding .name is the name of the format and through that we control the structure of the output.encoding is the encoding used for the structure created by the converter. Currently, ajson encoding is supported. More encoders are planned in future (e.g. Avro, Parquet, ...)

opencdc/json

This is the default format. An extensive description of this format is availablehere .

Options

None.

debezium/json

Generates output compatible with Debezium connectors.

Options

debezium.schema.name

(Optional)

Name of schema to be used in converted Debezium records.

Default value : none

debezium.rawData.key

(Optional)

If the record to be converted doesn't contain a structured payload or its payload cannot be parsed as JSON, then this key defines the field into which the "raw" payload data will be put.

Default value :opencdc.rawData

template

The user gets full control over the output format by using Go templates . The value provided to the template is a sdk.Record , so the template has access to all its fields (e.g..Position ,.Key ,.Metadata , and so on). We also inject all template functions provided by sprig to make it easier to write templates.

Options

The only option for this formatter is Go template itself, as described here .

Examples

Example 1: A pipeline with a Debezium record formatter

The following pipeline has a generator source and a file destination. We use thedebezium/json record formatter so that the file destination outputs the records in the Debezium format. Additionally, we set the schema name to beemployee by setting the format options todebezium.schema.name=employee .

---

version :

2.2

pipelines : -

id : pipeline1 status : running name : pipeline1 description :

'A pipeline with a Debezium record formatter' connectors : -

id : source1 type : source plugin : builtin : generator name : source1 settings : recordCount :

"10" readTime :

"1s" format.options :

"name:string,company:string" format.type :

"structured"

id : destination1 type : destination plugin :

"builtin:file" name : destination1 settings : sdk.record.format :

"debezium/json" sdk.record.format.options :

'debezium.schema.name=employee' path : /tmp/file - destination.txt If you run this pipeline andtail the destination file, you'll see something like this:

{ "schema" :

{ "type" :

"struct" , "name" :

"employee" , "fields" :

[// omitted for brevity] } , "payload" :

{ "before" :

null , "after" :

{ "company" :

"string 3cea28f1-8ec2-4035-b4d3-790a9239feb3" , "name" :

"string 8fbbbf0c-aaaa-499f-b145-ac5345c8c067" } , "source" :

{ "conduit.source.connector.id" :

"pipeline1:source1" , "opencdc.readAt" :

"1680796794653155968" , "opencdc.version" :

"v1" } , "op" :

"c" , "ts_ms" :

1680796794653 , "transaction" :

null } }

Example 2: A pipeline with a template record formatter

The following pipeline has a generator source and a file destination. The generator source produces records which have aname field and acompany field. In this pipeline, we use thetemplate record formatter to extract only theid field and then prefix it withemployee ID .

---

version :

2.2

pipelines : -

id : pipeline1 status : running name : pipeline1 description :

'A pipeline with a template record formatter' connectors : -

id : source1 type : source plugin : builtin : generator name : source1 settings : recordCount :

"3" readTime :

"1s" format.options :

"id:int,name:string,company:string" format.type :

"structured"

id : destination1 type : destination plugin :

"builtin:file" name : destination1 settings : sdk.record.format : template

print only the payload, format it as a string (normally it's a byte slice)

sdk.record.format.options :

'{{ printf "employee ID %d" .Payload.After.id }}' path : /tmp/file - destination.txt If you run Conduit and thentail the destination file, you'll see something like this:

employee ID 636561854 employee ID 1848322815 employee ID 123675258Edit this page Previous Building Connectors Next Connector Lifecycle