Cloudera Runtime 7.1.9

Integrating with Schema Registry

Date published: 2019-08-22
Date modified: 2023-09-07

https://docs.cloudera.com/

Legal Notice

Contents

Integrating Schema Registry with NiFi

When using NiFi to convert events from one file format to another, you need to integrate Schema Registry with NiFi because NiFi fetches the schema for the event from Schema Registry to be able to perform the conversion.

NiFi record-based Processors and Controller Services

Learn about the NiFi Processors and Controller Services that facilitate converting events from one file format to another.

The RecordReader and RecordWriter Processors and Controller Services allow you to convert events from one file format (JSON, XML, CSV, Avro) to another (JSON, XML, CSV, Avro). These Controller Services use Schema Registry to fetch the schema for the event to perform this conversion.

NiFi includes the following RecordReader and RecordWriter Processors:

- ConsumeKafkaRecord_0_10 1.2.0
- ConvertRecord
- PublishKafkaRecord_0_10
- PutDatabaseRecord
- QueryRecord
- SplitRecord

NiFi also includes the following record-based Controller Services:

- HortonworksSchemaRegistry
- AvroRecordSetWriter
- CSVRecordSetWriter
- FreeFormTextRecordSetWriter
- JsonRecordSetWriter
- ScriptedRecordSetWriter

Configuring Schema Registry instance in NiFi

Configuring Schema Registry in NiFi is essential for seamless communication between the two components. To enable Schema Registry to communicate with NiFi dataflows, you need to configure the HortonworksSchemaRegistry Controller Service with the Schema Registry instance URL.

Before you begin

You have already installed Schema Registry.

Procedure

1. Navigate to Controller Settings in the Global Menu and select the Controller Services tab.
2. Click the + icon to access the Add Controller Service dialog.
3. Search for HortonworksSchemaRegistry in the filter box and click Add.
4. Open the Configure Controller Service dialog by clicking the Edit icon.
5. Provide the Schema Registry URL with which you want NiFi to communicate and click Apply.
6. Enable HortonworksSchemaRegistry by clicking the Enable icon, selecting the Scope, and then clicking Enable again.

Setting schema access strategy in NiFi

Learn how to add Controller Services in NiFi globally or per Process Group and then how to configure them so that they can fetch schemas from Schema Registry.

You can configure Controller Services either globally, before you have created a Process Group, or at any time, on a per Process Group basis.

Adding Controller Services globally

1. To access the Controller Services configuration dialog for gobal configuration, click the Global Menu at the top right of your canvas, and select Controller Settings.
2. Click the + icon to display the NiFi Settings dialog.
3. Use the Filter box to search for the Controller Service you want to add, select that service, and click Add.

Adding Controller Services per Process Group

1. Click on your Process Group, and then right-click anywhere on your canvas.
2. Click Configure to display the Process Group Configuration dialog.
3. Click the Controller Services tab, and then click + to display the Add Controller Service dialog.
4. Use the Filter box to search for the Controller Service you want to add, select that service, and click Add.

Configuring record reader and writer Controller Services for integration with Schema Registry

1. From the Process Group Configuration view, click the Edit icon from the right-hand column. The Configure Controller Service dialog appears.
2. Click the Properties tab.
3. The *Schema Access Strategy* specifies how to obtain the schema to be used for interpreting FlowFile data. To ensure integration with Schema Registry, configure *Schema Access Strategy* with one of the following two values:
   - HWX Schema Reference Attributes

     The NiFi FlowFile is given a set of 3 attributes to describe the schema:

     - schema.identifier
     - schema.version
     - schema.protocol.version
   - HWX Content-Encoded Schema Reference

     Each NiFi FlowFile contains a reference to a schema stored in Schema Registry. The reference is encoded as a single byte indicating the protocol version, 8 bytes indicating the schema identifier and 4 bytes indicating the schema version.
4. The *Schema Write Strategy* specifies how the schema for a record should be added to FlowFile data. To ensure integration with Schema Registry, configure *Schema Write Strategy* with either *HWX Schema Reference Attributes* or *HWX Content-Encoded Schema Reference*.

Adding and configuring record-enabled Processors

To streamline your dataflows and improve overall performance, you can use record-enabled RecordReader and RecordWriter Processors.

About this task

Record-enabled Processors allow you to convert data between formats by specifying Controller Services for record reading and record writing. To add and configure record-enabled Processors, perform the following tasks:

Procedure

1. From the NiFi UI, drag the Processor icon onto your canvas to display the Add Processor dialog.
2. Use the Filter box to find the processor you want to add. Available record-enabled processors are:
   - ConsumeKafkaRecord_0_10
   - ConvertRecord
   - PublishKafkaRecord_0_10
   - PutDatabaseRecord
   - QueryRecord
   - SplitRecord
3. Select the processor you want, and click Add.
4. Right-click the processor on the canvas, and select Configure to display the Configure Processor dialog.
5. Click the Properties tab and select a Controller Service value for the Record Reader and Record Writer values.
6. Click OK, and then Apply.

Integrating Schema Registry with Kafka

Learn the manual configuration steps required to integrate Kafka and Schema Registry if you are running CDP without NiFi.

If you are running CDP without NiFi, integrate your Kafka producer and consumer manually. To do this, add a dependency on the Schema Registry SerDes and update the Kafka producer and Kafka consumer configuration files.

Adding a Schema Registry SerDes dependency

Add the following text to schema-registry-serdes:

```
<dependency>
      <groupId>com.hortonworks.registries</groupId>
      <artifactId>schema-registry-serdes</artifactId>
</dependency>
```

Integrating the Kafka producer

1. Add the following text to the Kafka producer configuration:

```
config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
config.putAll(Collections.singletonMap(SchemaRegistryClient.Configuration
.SCHEMA_REGISTRY_URL.name(), props.get(SCHEMA_REGISTRY_URL)));
config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.c
lass.getName());
config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, KafkaAvroSeria
lizer.class.getName());
```

2. Edit the above text with values for the following properties:
   - schema.registry.url
   - key.serializer
   - value.serializer

Integrating the Kafka consumer

1. Add the following text to the Kafka consumer configuration:

```
config.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
```

```
config.putAll(Collections.singletonMap(SchemaRegistryClient.Configuration
.SCHEMA_REGISTRY_URL.name(), props.get(SCHEMA_REGISTRY_URL)));
config.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializ
er.class.getName());
config.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, KafkaAvroDes
erializer.class.getName());
```

2. Edit the above text with values for the following properties:
   - schema.registry.url
   - key.deserializer
   - value.deserializer

Customizing client cache duration

By default, Schema Registry clients are only able to cache the schema for 5 minutes. To customize the cache duration, configure the cache properties.

Customize the values for the following properties:

- Schema Metadata Cache

  ```
  schema.registry.client.schema.metadata.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in schema metadata cache. Default value is 300.
- Schema Version Cache

  ```
  schema.registry.client.schema.version.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in schema version cache. Default value is 300.
- Schema Text Cache

  ```
  schema.registry.client.schema.text.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in schema text cache. Default value is 300.
- Class Loader Cache

  ```
  schema.registry.client.class.loader.cache.expiry.interval.secs
  ```

  Expiry interval (in seconds) of an entry in the serializer/deserializer class loader cache. Default value is 300.

Integrating Schema Registry with Flink and SSB

When using Kafka with Schema Registry, you can integrate them with Flink for large-scale data streams processing, and with SSB to import schema as virtual tables and use the tables to create queries on streams of data through SQL.

When Kafka is chosen as source and sink for your application, you can use Cloudera Schema Registry to register and retrieve schema information of the different Kafka topics. You must add Schema Registry dependency to your Flink project and add the appropriate schema object to your Kafka topics. Additionally, you can add Schema Registry as a catalog to SQL Stream Builder (SSB) to import existing schemas for use with SQL queries.

You can integrate Flink with your project along with Kafka and Schema Registry to process data streams at a large scale and to deliver real-time analytical insights about your processed data. For more information about how to integrate Flink with Schema Registry, see *Schema Registry with Flink*.

You can integrate Schema Registry as a catalog with SSB to import schema as virtual tables which you can use to create queries on the streams of data through SQL. For more information about how to integrate SSB with Schema Registry, see *Adding Catalogs*.

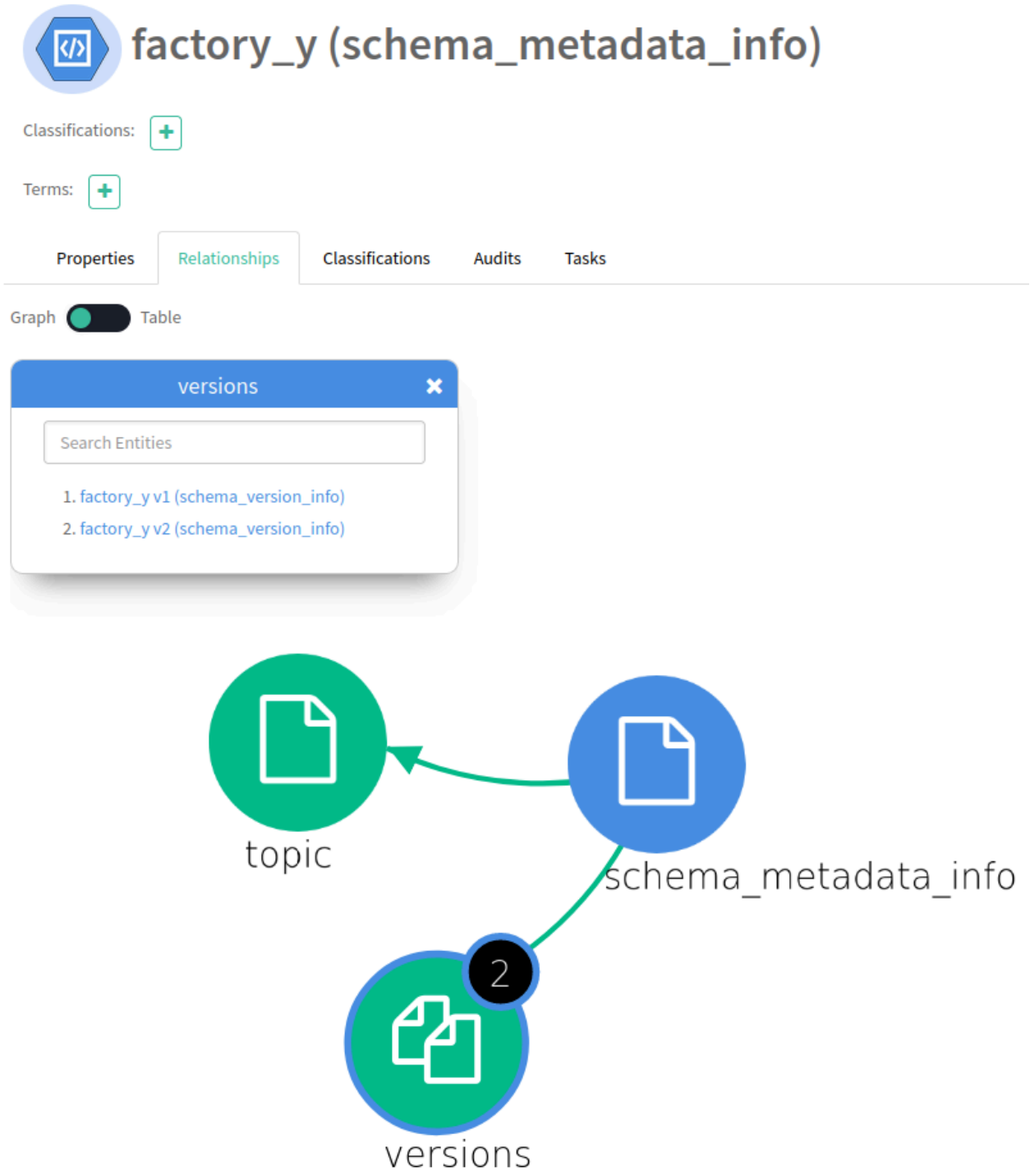
Related Information
Schema Registry with Flink
Adding Catalogs

Integrating Schema Registry with Atlas

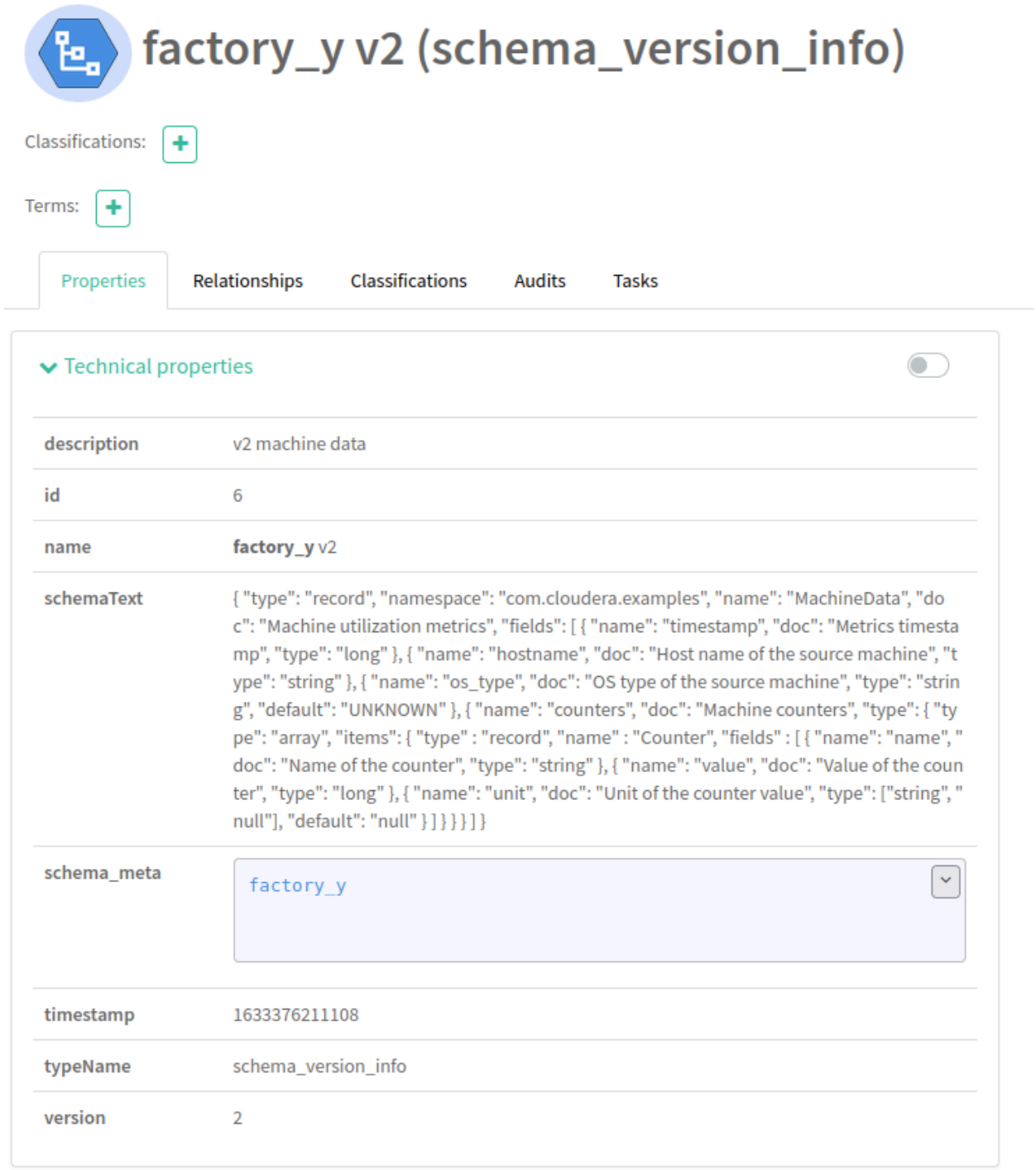
If you want to view schemas in the Atlas UI, you need to integrate Schema Registry with Apache Atlas.

When you integrate Schema Registry with Atlas, the relationship between the schema, topic, and all versions of a schema can be explored with ease in Atlas.

The following image shows a sample relationship:



You can select a schema version to view the schema text in JSON.



For more information, see *Schema Registry metadata collection*.

Improving performance in Schema Registry

Adding Knox as a load balancer can increase the performance of Schema Registry.

You can increase the performance of Schema Registry by adding Apache Knox, the CDP load balancer, to your Schema Registry configuration. This increase in performance offsets the minor decrease in performance caused by the lack of server-side caching.

By default, Schema Registry does not have server-side caching. Each time a client sends a query, the client reads the data directly from the database.

The lack of server-side caching creates a minor performance impact. However, it enables Schema Registry to be stateless and be distributed over multiple hosts. All hosts use the same database. This enables a client to insert a schema on host A and query the same schema on host B.

You can store JAR files similarly in HDFS and retrieve a file even if one of the hosts is down.

To make Schema Registry highly available in this setup, include a load balancer on top of the services. Clients communicate through the load balancer. The load balancer then determines which host to send the requests to. If one host is down, the request is sent to another host. In CDP, the load balancer is Apache Knox.

The following diagram shows the flow of a client query and response: