Informatica® Data Engineering Streaming
10.5.1

User Guide

Informatica Data Engineering Streaming User Guide
10.5.1
September 2021

Table of Contents

Preface

Use the *Informatica® Data Engineering Streaming User Guide* to learn about the configuration, guidelines, and usage of streaming mappings on the Spark engine in a non-native environment. The guide also includes information about the transformation processing differences and the sources and targets that you can use in streaming mappings.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

Introduction to Data Engineering Streaming

This chapter includes the following topics:

Data Engineering Streaming Overview

Use Informatica Data Engineering Streaming to prepare and process streams of data in real time and uncover insights in time to meet your business needs. Data Engineering Streaming provides pre-built connectors such as Kafka, Amazon Kinesis, HDFS, enterprise messaging systems, and data transformations to enable a code-free method of defining data integration logic.

Data Engineering Streaming builds on the best of open source technologies. It uses Spark Structured Streaming for stream processing, and supports other open source stream processing platforms and frameworks, such as Kafka and Hadoop. Spark Structured Streaming is a scalable and fault-tolerant open source stream processing engine built on the Spark engine.

You can create streaming mappings to stream machine, device, and social media data in the form of messages. Streaming mappings collect machine, device, and social media data in the form of messages. The mapping builds the business logic for the data and pushes the logic to the Spark engine for processing. Use a Messaging connection to get data from Apache Kafka brokers, Amazon Kinesis, and Azure Event Hubs.

The Spark engine runs the streaming mapping continuously. The Spark engine reads the data, divides the data into micro batches, processes it, updates the results to a result table, and then writes to a target.

You can stream the following types of data:

- Application and infrastructure log data
- Change data(CDC) from databases
- Clickstreams from web servers
- Geo-spatial data from devices
- Sensor data

- Time series data

- Supervisory Control And Data Acquisition (SCADA) data

- Message bus data

- Programmable logic controller (PLC) data

- Point of sale data from devices

You can stream data to different types of targets, such as Kafka, HDFS, Amazon Kinesis Firehose, Amazon S3, HBase tables, Hive tables, JDBC-compliant databases, Microsoft Azure Event Hubs, and Azure Data Lake Store.

Data Engineering Streaming works with Data Engineering Integration to provide streaming capabilities. In a Hadoop environment, Data Engineering Streaming uses YARN to manage the resources on a Spark cluster. It uses third-party distributions to connect to and push job processing to a Hadoop environment. In a Databricks environment, Data Engineering Streaming uses the built-in standalone resource manager to manage the Spark clusters.

Use Informatica Developer (the Developer tool) to create streaming mappings. To run the streaming mapping, you can use the Hadoop run-time environment and the Spark engine or the Databricks run-time environment and the Databricks Spark engine for the supported sources and targets.

You can configure high availability to run the streaming mappings on the Hadoop or Databricks cluster.

For more information about running mappings on the Spark engine, see the *Data Engineering Integration User Guide*.

Streaming Process

A streaming mapping receives data from unbounded data sources. An unbounded data source is one where data is continuously flowing in and there is no definite boundary. Sources stream data as events. The Spark engine processes the data and continuously updates the results to a result table.

The following image shows how the Spark engine receives data and publishes data in micro batches:
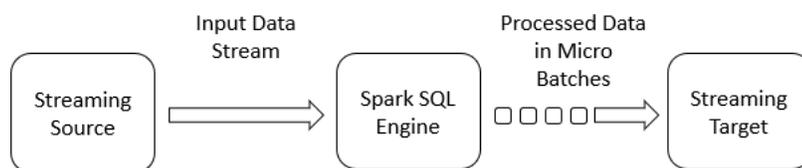
The Spark engine uses Spark Structured Streaming to process data that it receives in batches. Spark Structured Streaming receives data from streaming sources such as Kafka and divides the data into micro batches. The Spark engine continuously processes data streams as a series of small batch jobs with end-to-end latencies as low as 100 milliseconds with exactly-once fault tolerance guarantees.

For more information about Spark Structured Streaming, see the Apache Spark documentation at https://spark.apache.org/documentation.html.

You can perform the following high-level tasks in a streaming mapping:

1. Identify sources from which you need to stream data. You can access data that is in XML, JSON, Avro, flat, or binary format.
   In Hadoop environment, you can use Kafka Amazon Kinesis stream and Azure Event Hubs sources to connect to multiple data engineering sources.

2. Configure the mapping and mapping logic to transform the data.

3. Run the mapping on the Spark engine in the Hadoop environment or on the Databricks Spark Engine in the Databricks environment.

4. Write the data to Kafka targets, HDFS complex files, HBase, Azure Event Hubs, Amazon S3, Azure Data Lake Storage, JMS, and Kinesis Firehose delivery streams.

5. Monitor the status of your processing jobs. You can view monitoring statistics for your processing jobs in the Monitoring tool.

Component Architecture

The Data Engineering Streaming components for a streaming mapping include client tools, application services, repositories, and third-party tools.

The following image shows the components that Data Engineering Streaming uses for streaming mappings:
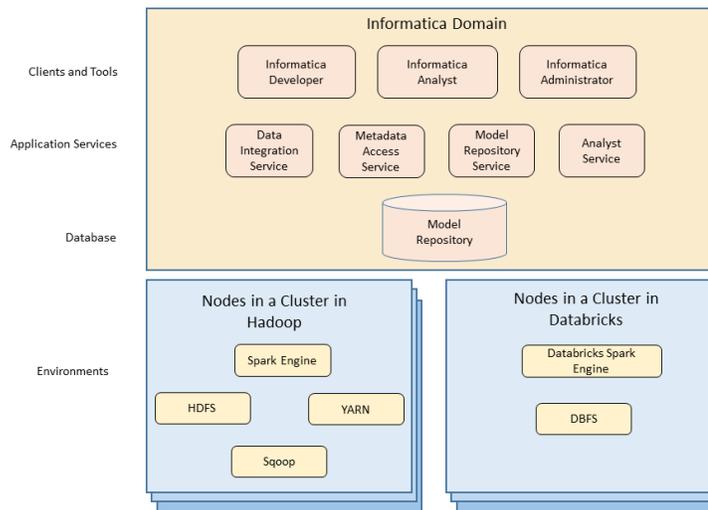
Clients and Tools

Based on your product license, you can use multiple Informatica tools and clients to manage streaming mappings.

Use the following tools to manage streaming mappings:
Informatica Administrator

Monitor the status of mappings on the Monitoring tab of the Administrator tool. The Monitoring tab of the Administrator tool is called the Monitoring tool.

Informatica Developer

Create and run mappings on the Spark engine or the Databricks Spark engine from the Developer tool.

Informatica Analyst

Create rules in Informatica Analyst and run the rules as mapplets in a streaming mapping.

Application Services

Data Engineering Streaming uses application services in the Informatica domain to process data. The application services depend on the task you perform.

Data Engineering Streaming uses the following application services when you create and run streaming mappings:

Data Integration Service

In a Hadoop environment, the Data Integration Service processes mappings on the Spark engine. The Data Integration Service retrieves metadata from the Model repository when you run a Developer tool mapping. The Developer tool connects to the Data Integration Service to run mappings.

In a Databricks environment, when you run a job on the Databricks Spark engine, the Data Integration Service pushes the processing to the Databricks cluster, and the Databricks Spark engine runs the job.

Metadata Access Service

The Metadata Access Service is a user-managed service that provides metadata from a Hadoop cluster to the Developer tool at design time. HBase, HDFS, Hive, and MapR-DB connections use the Metadata Access Service when you import an object from a Hadoop cluster. Create and configure a Metadata Access Service before you create HBase, HDFS, Hive, MapR Streams, and MapR-DB connections.

Model Repository Service

The Model Repository Service manages the Model repository. The Model Repository Service connects to the Model repository when you run a mapping.

Analyst Service

The Analyst Service runs the Analyst tool in the Informatica domain. The Analyst Service manages the connections between service components and the users that have access to the Analyst tool.

Repository

Data Engineering Streaming includes a repository to store data related to connections and source metadata. Data Engineering Streaming uses application services in the Informatica domain to access data in the repository.

Data Engineering Streaming stores structured streaming mappings in the Model repository. You can manage the Model repository in the Developer tool.

Third-Party Applications

Data Engineering Streaming uses third-party distributions to connect to a Spark engine on a Hadoop cluster or to a Databricks Spark engine on a Databricks cluster.

In a Hadoop environment, Data Engineering Streaming pushes job processing to the Spark engine. It uses YARN to manage the resources on a Spark cluster.

In a Databricks environment, Data Engineering Streaming pushes job processing to the Databricks cluster, and the Databricks Spark engine runs the job.

Data Engineering Streaming Engines

When you run a streaming mapping, you can choose to run the mapping in a Hadoop environment or in a Databricks environment. When you validate a mapping, you can validate it against one or all of the engines. The Developer tool returns validation messages for each engine.

When you run a streaming mapping, the Data Integration Service uses a proprietary rule-based methodology to determine the best engine to run the mapping. The rule-based methodology evaluates the mapping sources and the mapping logic to determine the engine. The Data Integration Service translates the mapping logic into code that the engine can process, and it transfers the code to the engine.

For more information about the run-time process on the different engines, see *Data Engineering Integration User Guide*.

Example

You run the IT department of a major bank that has millions of customers. You want to monitor network activity in real time. You need to collect network activity data from various sources such as firewalls or network devices to improve security and prevent attacks. The network activity data includes Denial of Service (DoS) attacks and failed login attempts made by customers. The network activity data is written to Kafka queues.

You perform the following tasks:

1. Create a streaming mapping to read data from the Kafka queues that stream data in JSON, XML, CSV, or Avro formats.
2. Add a Lookup transformation to get data from a particular customer ID.
3. Add a Window transformation to accumulate the streamed data into data groups before processing the data.
4. Process the data. Add an Aggregator transformation to perform aggregations on the data from the customer ID.
5. Monitor jobs. Monitor statistics for the mapping job on the Monitoring tab of the Administrator tool.

The following image shows the mapping:
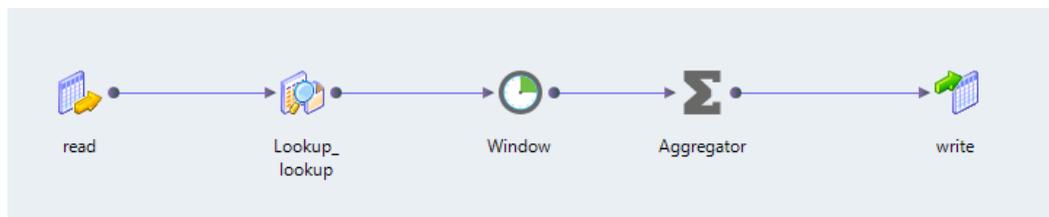
CHAPTER 2

Data Engineering Streaming Administration

This chapter includes the following topics:

Data Engineering Streaming Configuration Overview

After you integrate the Informatica domain with the Hadoop environment, you can use Data Engineering Streaming in conjunction with Data Engineering Integration.

Before you create streaming mappings perform the following tasks:

- Verify the Hadoop distribution versions that Data Engineering Streaming supports.
- Before you create and use JMS connections and JMS data objects in streaming mappings, complete the required tasks in "Configure Java Authorization and Authentication Service (JAAS) " on page 20.
- Before you read from or write to a Kerberized Kafka cluster, complete the required tasks in "Configuration for Kerberized Kafka Clusters" on page 19.

For more information about integration tasks, see the *Data Engineering Integration Guide*.

Authorization Systems

You can run streaming mappings on a Hadoop cluster that uses a supported security management system.

Hadoop clusters use a variety of security management systems for user authorization. The following table shows the Spark engines supported for the security management system installed on the Hadoop platform:

Hadoop Distribution	Apache Ranger	Apache Sentry	SQL Authorization
Amazon EMR	Not supported	Not supported	Not supported
Azure HDInsight	Supported	Not supported	Supported
Cloudera CDH	Not supported	Supported	Supported
Hortonworks HDP	Supported	Not supported	Supported
MapR	Not supported	Not supported	Not supported

Additional Information for Authorization

Consider the following additional security support statements:

- Informatica supports an Azure HDInsight cluster with Enterprise Security Package. The Enterprise Security Package uses SQL and Apache Ranger for authorization.
- Data Engineering Streaming supports one type of authorization for each distribution with the following exceptions:
    - Apache Ranger and SQL authorization are both supported on an instance of Hortonworks HDP.
    - Apache Sentry and SQL authorization are both supported on an instance of Cloudera CDH.

For more information about authorization systems, see *Data Engineering Administrator Guide*.

Authentication Systems on Hadoop

You can run streaming mappings on a Hadoop cluster that uses a supported security management system.

Hadoop clusters use a variety of security management systems for user authentication. The following table shows the Spark engine support for the security management system installed on the Hadoop platform:

Hadoop Distribution	HDFS TE	Kerberos	LDAP	SSL/TLS
Amazon EMR	Not supported	Supported	Not supported	Not supported
Azure HDInsight	Not supported	Not supported	Not supported	Not supported

Hadoop Distribution	HDFS TE	Kerberos	LDAP	SSL/TLS
Cloudera CDH	Supported	Supported	Supported	Supported
Hortonworks HDP	Supported	Supported	Supported	Supported
MapR	Not supported	Supported	Not supported	Not supported

Additional Information for Authentication

Consider the following additional security support statements:

- Apache Knox is not available in any of the mentioned Hadoop distributions.
- Sqoop cannot access Kerberos or SSL enabled databases.

For more information about authentication systems, see *Data Engineering Administrator Guide*.

Authentication Systems on Databricks

The Data Integration Service uses token-based authentication to provide access to the Databricks environment.

The Databricks administrator creates a token user and generates tokens for the user. The Databricks cluster configuration contains the token ID required for authentication.

Note: If the token has an expiration date, verify that you get a new token from the Databricks administrator before it expires.

Configuration for Kerberized Kafka Clusters

To read from or write to a Kerberized Kafka cluster, configure the default realm, KDC, Hadoop connection properties, and Kafka data object read or write data operation properties.

Configure the Default Realm

Before you read from or write to a Kerberized Kafka cluster, perform the following tasks:

1. Ensure that you have the krb5.conf file for the Kerberized Kafka server.
2. Configure the default realm and KDC. If the default `/etc/krb5.conf` file is not configured or you want to change the configuration, add the following lines to the `/etc/krb5.conf` file:

   ```
   [libdefaults]
   default_realm = <REALM NAME>
   ```

```
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true

[realms]
<REALM NAME> = {
kdc = <Location where KDC is installed>
admin_server = <Location where KDC is installed>
                                          }
 [domain_realm]
.<domain name or hostname> = <KERBEROS DOMAIN NAME>
<domain name or hostname> = <KERBEROS DOMAIN NAME>
```

Configure Java Authorization and Authentication Service (JAAS)

To pass a static JAAS configuration file into the JVM using the `java.security.auth.login.config` property at run time, perform the following tasks:

Use a Static JAAS Configuration File

1. Ensure that you have JAAS configuration file.
    For information about creating JAAS configuration and configuring Keytab for Kafka clients, see the
    Apache Kafka documentation at https://kafka.apache.org/0101/documentation/#security

    For example, the JAAS configuration file can contain the following lines of configuration:
    ```
    //Kafka Client Authentication. Used for client to kafka broker connection
    KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule required
    doNotPrompt=true
    useKeyTab=true
    storeKey=true
    keyTab="<path to keytab  file>/<keytab file name>"
    principal="<principal name>"
    client=true
    };
    ```

2. Place the JAAS config file and keytab file in the same location on all the nodes of the Hadoop cluster.
    Put the files in a location that is accessible to all nodes on the cluster, such as `/etc` or `/temp`.

    On the **Spark Engine** tab of the Hadoop connection properties, update the `extraJavaOptions` property of
    the executor and the driver in the **Advanced Properties** property. Click **Edit** and update the properties in
    the following format:
    ```
    spark.executor.extraJavaOptions=-Djava.security.egd=file:/dev/./urandom
    -XX:MaxMetaspaceSize=256M -Djavax.security.auth.useSubjectCredsOnly=true
    -Djava.security.krb5.conf=/<path to krb5.conf file>/krb5.conf
    -Djava.security.auth.login.config=/<path to jAAS config>/<kafka_client_jaas>.config

    spark.driver.cluster.mode.extraJavaOptions=-Djava.security.egd=file:/dev/./urandom
    -XX:MaxMetaspaceSize=256M -Djavax.security.auth.useSubjectCredsOnly=true
    -Djava.security.krb5.conf=/<path to krb5.conf file>/krb5.conf
    -Djava.security.auth.login.config=<path to jaas config>/<kafka_client_jaas>.config
    ```

3. Configure the following properties in the data object read or write operation:

    *   Data object read operation. Configure the **Consumer Configuration Properties** property in the
        advanced properties.

    *   Data object write operation. Configure the **Producer Configuration Properties** property in the
        advanced properties.

    Specify the following value:
    ```
    security.protocol=SASL_PLAINTEXT,sasl.kerberos.service.name=kafka,sasl.mechanism=GSSA
    PI
    ```

Embed the JAAS Configuration

To embed the JAAS configuration in the `sasl.jaas.config` configuration property, perform the following tasks:

1. On the **Spark Engine** tab of the Hadoop connection properties, update the `extraJavaOptions` property of the executor and the driver in the **Advanced Properties** property. Click **Edit** and update the properties in the following format:

   ```
   spark.executor.extraJavaOptions=-Djava.security.egd=file:/dev/./urandom
   -XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500 -
   Djava.security.krb5.conf=<path to krb5.conf file>


   spark.driver.cluster.mode.extraJavaOptions=-Djava.security.egd=file:/dev/./urandom
   -XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500 -
   Djava.security.krb5.conf=<path to krb5.conf file>
   ```

2. Configure the following properties in the data object read or write operation:

   - Data object read operation. Configure the **Consumer Configuration Properties** property in the advanced properties.

   - Data object write operation. Configure the **Producer Configuration Properties** property in the advanced properties.

   Specify the following value:

   ```
   security.protocol=SASL_PLAINTEXT,sasl.kerberos.service.name=kafka,sasl.mechanism=GSSA
   PI,sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   storeKey=true doNotPrompt=true serviceName="<service_name>" keyTab="<location of
   keytab file>"
   client=true principal="<principal_name>";
   ```

Configuration for Amazon Kinesis

To create an Amazon Kinesis connection, create AWS credentials for the users. AWS credentials consist of access key ID and secret access key. Use the Amazon Kinesis connection to access Amazon Kinesis Streams as sources or Amazon Kinesis Firehose as targets.

Create AWS Credentials

Create AWS credentials for the users, such as access key ID and secret access key. An authentication type can be selected by the user during the creation of an Amazon Kinesis connection, such as AWS credential profile and cross-account IAM role. The default authentication type is AWS credential profile.

Access Key and ID

Generate an Access Key ID and Secret Access Key for the users in AWS.

AWS Credential Profile

You can define AWS credential profiles in the credentials file. Each credential profile consists of secret access key and access key ID.

Users can use the AWS credential profile names to use different AWS credentials at run time than the AWS credentials that they specify when they create an Amazon Kinesis connection with an Amazon Kinesis Streams as a source and Amazon Kinesis Firehose as a target.

Place the credentials file in the same location on all the cluster nodes. Default location is `<yarn home directory>/.aws/credentials`.

Note: AWS credential profile is used at run time only.

Cross-account IAM Role

You can create an IAM role to allow a user of an AWS account to access resources of another AWS account. IAM roles allow you to define a set of permissions to access the AWS resources.

Users can share resources in one AWS account with users in a different AWS account without the need to create individual IAM users in each AWS account.

Note: Cross-account IAM role is not supported for Amazon Kinesis Firehose as target.

Configuration for Amazon Kinesis Streams Sources

To use Amazon Kinesis Streams as sources, grant required permissions to the user. To use cross-account IAM role, create an IAM role and grant access to the role.

To configure access for Amazon Kinesis Streams as a source, perform the following tasks :

- Grant consumer permissions that are part of the IAM policy to the AWS credentials that the IAM user specifies in the access key id.
  For the list of permissions, see the AWS documentation at
  https://docs.aws.amazon.com/streams/latest/dev/learning-kinesis-module-one-iam.html
- Grant the following permissions to the user to fetch metadata:
  - kinesis:DescribeStream
  - kinesis:GetShardIterator
  - kinesis:GetRecords
- To use cross-account IAM role, create an IAM role in an AWS account. IAM roles are used to provide secure access to AWS resources. The role is used to establish a trusted relationship between various AWS accounts. Additional restrictions are enforced using an external ID and the IAM role can only be assumed by using the external ID if the external ID is specified for a cross-account IAM role.

  Note: Multi-factor authentication is not supported.

Configuration for Amazon Kinesis Firehose Targets

To use Amazon Kinesis Firehose as targets create an AWS account, grant the required permissions and privileges to the IAM role.

To configure access for Amazon Firehose as a target, perform the following tasks:

- Create an AWS account with the required IAM permissions for the IAM user to use the AWS services such as, Kinesis Firehose, S3, Redshift, and Elastic Search.
- Grant Redshift INSERT privilege to the IAM user if the user wants to copy data from the Amazon S3 bucket to the Redshift cluster.
- Define a Firehose Delivery Stream with either S3, Redshift or Elastic Search as its destination. Configure source as Direct PUT or other sources.
- Grant required permissions to the IAM user credentials based on the target the user is writing to.
  For a list of permissions, see the AWS documentation at
  https://docs.aws.amazon.com/firehose/latest/dev/controlling-access.html#access-to-firehose

For more information about Amazon Kinesis connection, see "Amazon Kinesis Connection" on page 173.

Configuration for Azure Event Hubs

Before you create an Azure Event Hub connection, verify the following prerequisites:

1. You must have a Microsoft Azure account with a minimum role of contributor.
2. Verify that you have an Active Directory application.
3. Verify that your Active Directory application has permissions for the following API and directory:
   - Windows Azure Service Management API
   - Windows Azure Active Directory
4. Verify that the Azure Active Directory application is added with a reader role to your Azure account subscription.
5. Verify that you have an Event Hub Namespace.

For more information about Azure Event Hub connection, see .

Configuration for Snowflake Targets

To use Snowflake as targets in streaming mappings, complete the following prerequisites:

- Verify that you have the required permissions to access the Snowflake table and the insert privileges to the table.
- Verify that you have the privileges to create the Snowflake named internal stages and the Snowflake pipes.
- Generate the PEM key for authentication to Snowflake. For more information about generating the PEM keys, see the Snowflake documentation at the following website:
  https://docs.snowflake.net/manuals/LIMITEDACCESS/spark-connector-streaming.html

CHAPTER 3

Sources in a Streaming Mapping

This chapter includes the following topics:

Sources in a Streaming Mapping Overview

You can access log file data, sensor data, Supervisory Control And Data Acquisition (SCADA) data, message bus data, Programmable logic controller (PLC) data on the Spark SQL engine in the Hadoop environment or Databricks Spark SQL engine in the Databricks environment.

You can create physical data objects to access the different types of data. Based on the type of source you are reading from, you can create the relevant data objects.

Sources in a Streaming Mapping on Hadoop and Databricks

A streaming mapping that runs in the Hadoop environment can include file, database, and streaming sources. A streaming mapping that runs in the Databricks environment can include streaming sources.

The following table lists the physical data objects that you can create based on the type of source you read from in either Hadoop or Databricks environment:

Sources in a Streaming Mapping	Environment
Amazon Kinesis	Hadoop, Databricks
Azure Event Hubs	Hadoop, Databricks
Confluent Kafka	Hadoop, Databricks
Google PubSub	Hadoop
HBase	Hadoop
JMS	Hadoop
Kafka	Hadoop, Databricks
MapR Streams	Hadoop

Note: Intelligent structure model is supported for Amazon Kinesis, Azure Event Hubs, Confluent Kafka, and Kafka data objects for streaming mappings that run on the Databricks Spark engine.

You can run streaming mappings in the Databricks environment on the AWS or Azure platforms. The following table shows the list of sources that you can include in a streaming mapping based on the cloud platform:

Sources	Cloud Platform
Amazon Kinesis	AWS
Azure Event Hubs	Azure
Confluent Kafka	AWS, Azure
Kafka	AWS, Azure

Processing Hierarchical Data in Streaming Mappings

Data objects in a streaming mapping can process hierarchical data through complex data types. If the source contains hierarchical data, you must enable the read data operation to project columns as complex data types.

The following are the complex data types:

- Array. An array is an ordered collection of elements.
- Map. A map is an unordered collection of key-value pair elements.
- Struct. A struct is a collection of elements of different data types.

The following table shows the format and complex data types that sources in a streaming mapping support:

Object Type	File Type	Supported Formats
Amazon Kinesis Streams	Avro Binary Flat JSON XML	Flat Hierarchical Binary
Azure Event Hub	Avro Binary Flat JSON XML	Flat Hierarchical Binary
Confluent Kafka	Avro Binary Flat JSON XML	Flat Hierarchical Binary
JMS	Binary Flat JSON	Flat Hierarchical Binary
Kafka	Avro Binary Flat JSON XML	Flat Hierarchical Binary
MapR Streams	Avro Binary JSON XML	Flat Hierarchical Binary

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Amazon Kinesis Data Objects

An Amazon Kinesis data object is a physical data object that represents data in a Amazon Kinesis Data Stream. After you create an Amazon Kinesis connection, create an Amazon Kinesis data object to read from Amazon Kinesis Data Streams.

Kinesis Data Streams is a real-time data stream processing option that Amazon Kinesis offers within the AWS ecosystem. It is a customizable option for users who want to build custom applications to process and analyze streaming data. You must manually provision enough capacity to meet system needs.

When you configure the Amazon Kinesis data object, specify the name of the Amazon Kinesis Data Stream that you read from. After you create the data object, create a read operation to read data from an Amazon Kinesis Data Stream. You can then add the data object read operation as a source in streaming mappings.

When you configure the data operation properties, specify the format in which the data object reads data. When you read from Amazon Kinesis Data Stream sources, you can read data in JSON, XML, Avro, Flat, or binary format. When you specify XML format, you must provide a XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON or Flat format, you must provide a sample file.

You can associate the data object with an intelligent structure model and directly parse input from CSV, XML, Avro, or JSON input files.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

Note: You cannot run a mapping with an Amazon Kinesis data object on a MapR distribution.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

For more information about Kinesis Data Streams, see the Amazon Web Services documentation.

Related Topics:

-

Amazon Kinesis Data Object Overview Properties

Overview properties include general properties that apply to the Amazon Kinesis data object. The Developer tool displays overview properties of the data object in the **Overview** view.

You can configure the following overview properties for Amazon Kinesis data objects:

General

You can configure the following general properties for the Amazon Kinesis data object:

- Name. Name of the Amazon Kinesis operation.
- Description. Description of the Amazon Kinesis data object.
- Native Name. Name of the Amazon Kinesis data object.
- Path Information. The path of the data object in Amazon Kinesis. For example, `/DeliveryStreams/router1`

Column

> You can configure the name, native name, data type, precision, access type, scale, and description of the columns in the Amazon Kinesis resource.

Advanced

> The following are the advanced properties for the Amazon Kinesis data object:
>
> - Amazon Resource Name. The Kinesis resource that the Amazon Kinesis data object is reading from or writing to.
>
> - Type. The type of delivery stream that the Amazon Kinesis data object is reading from or writing to. The delivery stream is either Kinesis Stream or Firehose DeliveryStream
>
> - Number of Shards. Specify the number of shards that the Kinesis Stream is composed of. This property is not applicable for Firehose DeliveryStream.

Amazon Kinesis Header Ports

An Amazon Kinesis data object contains a default header port that represents metadata associated with events.

An Amazon Kinesis data object contains the following header port:

timestamp

> Time at which an event is generated. You can accumulate streamed data into data groups and then process the data groups based on the timestamp values.

Amazon Kinesis Data Object Read Operation Properties

The Data Integration Service uses read operation properties when it reads data from Amazon Kinesis Streams.

General Properties

The Developer tool displays general properties for Amazon Kinesis sources in the **Read** view.

The following table describes the general properties for the Amazon Kinesis data object read operation:

Property	Description
Name	The name of the Amazon Kinesis read operation. You can edit the name in the **Overview** view. When you use the Amazon Kinesis stream as a source in a mapping, you can edit the name in the mapping.
Description	The description of the Amazon Kinesis data object read operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Amazon Kinesis stream sources:

Property	Description
Name	The name of the source header field.
Type	The native data type of the source.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Sources Properties

The sources properties list the resources of the Amazon Kinesis data object.

The following table describes the sources property that you can configure for Amazon Kinesis Streams sources:

Property	Description
Sources	The sources which the Amazon Kinesis data object reads from. You can add or remove sources.

Run-time Properties

The run-time properties include properties that the Data Integration Service uses when reading data from the source at run time.

The run-time property for Amazon Kinesis Stream source includes the name of the Amazon Kinesis connection.

Advanced Properties

The following table describes the advanced properties for Amazon Kinesis Stream sources:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Guaranteed Processing	Ensures that the mapping processes messages that the sources publish and delivers them to the targets at least once. In the event of a failure, there could be potential duplicates but the messages are processed successfully. If the external source or the target is not available, the mapping execution stops to avoid any data loss. By default, the Guaranteed Processing property is selected.
Degree of Parallelism	The number of processes that run in parallel within a shard. Specify a value that is less than or equal to the number of shards.

Column Projections Properties

The following table describes the columns projection properties that you configure for Amazon Kinesis Stream sources:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.
Enable Column Projection	Indicates that you use a schema to read the data that the source streams. By default, the data is streamed in binary format. To change the format in which the data is processed, select this option and specify the schema format.
Schema Format	The format in which the source processes data. You can select one of the following formats: - XML - JSON - Avro - Flat
Use Schema	Specify the XSD schema for the XML format, the sample JSON for the JSON format. Specify a .avsc file for the Avro format or a sample file for the Flat format.

Property	Description
Use Intelligent Structure Model	Displays the intelligent structure model associated with the complex file. You can select a different model. You can associate the data object with an intelligent structure model and directly parse input from CSV, XML, Avro, or JSON input files. **Note:** - If you disable the column projection, the intelligent structure model associated with the data object is removed. If you want to associate an intelligent structure model with the data object again, enable the column projection and click **Select Model**. - A streaming mapping fails due to schema drift when you use Avro schema with an intelligent structure model. For more information about intelligent structure models, see the *Data Engineering Integration User Guide*.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for sources with hierarchical data. Select this option if the source has hierarchical data. For more information about hierarchical data, see the *Data Engineering Integration User Guide*.

Configuring Schema for Flat Files

Configure schema for flat files when you configure column projection properties.

1. On the **Column Projection** tab, enable column projection and select the flat schema format.

   The page displays the column projection properties page.

2. On the column projection properties page, configure the following properties:

   - Sample Metadata File. Select a sample file.

   - Code page. Select the UTF-8 code page.

   - Format. Format in which the source processes data. Default value is Delimited. You cannot change it.

3. Click **Next**.

4. In the delimited format properties page, configure the following properties:

Property	Description
Delimiters	Specify the character that separates entries in the file. Default is a comma (,). You can only specify one delimiter at a time. If you select **Other** and specify a custom delimiter, you can only specify a single-character delimiter.
Text Qualifier	Specify the character used to enclose text that should be treated as one entry. Use a text qualifier to disregard the delimiter character within text. Default is No quotes. You can only specify an escape character of one character.

Property	Description
Preview Options	Specify the escape character. The row delimiter is not applicable as only one row is created at a time.
Maximum rows to preview	Specify the rows of data you want to preview.

5. Click **Next** to preview the flat file data object.

   If required, you can change the column attributes. The data type timestampWithTZ format is not supported.

6. Click **Finish**.

   The data object opens in the editor.

Azure Event Hubs Data Objects

An Azure Event Hubs data object is a physical data object that represents an event hub object in Microsoft Azure Event Hubs data streaming platform and event ingestion service. After you create an Azure Event hubs connection, create an Azure Event hubs data object to read from event data from event hub event.

Azure Event Hubs is a highly scalable data streaming platform and event ingestion service, that receives and processes events. Azure Event Hubs can process and store events or data produced by distributed software and devices.

When you configure the Azure Event Hubs data object, specify the name of the event hub that you read from. After you create the data object, create a read operation to read event data from an event hub. You can then add the data object read operation as a source in streaming mappings.

When you configure the data operation properties, specify the format in which the Azure Event Hubs data object reads data. You can specify XML, JSON, Avro, or flat as format. When you specify XML format, you must provide a .xsd file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON or flat format, you must provide a sample file.

You can associate the data object with an intelligent structure model and directly parse input from CSV, XML, Avro, or JSON input files.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Azure Event Hubs Data Object Overview Properties

Overview properties include general properties that apply to the Azure Event Hubs data object. The Developer tool displays overview properties of the data object in the **Overview** view.

You can configure the following overview properties for Azure Event Hubs data objects:

General

You can configure the following general properties for the Azure Event Hubs data object:

- Name. Name of the Azure Event Hubs operation.
- Description. Description of the Azure Event Hubs data object.
- Native Name. Name of the Azure Event Hubs data object.
- Path Information. The path of the data object in Azure Event Hubs. For example, `/EventHubs/avroevents`

Column

You can configure the name, native name, data type, precision, access type, scale, and description of the columns in the Azure Event Hubs resource.

Advanced

The following are the advanced properties for the Azure Event Hubs data object:

- Location. The location of the Azure Event Hubs.
- Date of Creation. The date of creation of the Azure Event Hubs.
- Partition Count. The number of partitions that the Event Hub has when you import the data object.

Azure Event Hubs Header Ports

An Azure Event Hubs data object contains default header port that represent metadata associated with events.

An Azure Event Hubs data object contains the following header ports:

partitionKey

Partition keys to separate events into different partitions. Events with same partition key are sent to the same partition on the event hub.

timestamp

Time at which an event is generated. You can accumulate streamed data into data groups and then process the data groups based on the timestamp values.

Azure Event Hubs Data Object Read Operation Properties

The Data Integration Service uses read operation properties when it reads data from an Azure Event Hubs.

General Properties

The Developer tool displays general properties for Azure Event Hubs sources in the **Read** view.

The following table describes the general properties for the Azure Event Hubs data object read operation:

Property	Description
Name	The name of the Azure Event Hubs data object read operation. You can edit the name in the **Overview** view. When you use the Azure Event Hubs as a source in a mapping, you can edit the name in the mapping.
Description	The description of the Azure Event Hubs data object read operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Azure Event Hubs sources:

Property	Description
Name	The name of the source header field.
Type	The native data type of the source.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Sources Properties

The sources properties list the resources of the Azure Event Hubs data object.

The following table describes the sources property that you can configure for Azure Event Hubs events:

Property	Description
Sources	The sources which the Azure Event Hubs data object reads from. You can add or remove sources.

Run-time Properties

The run-time properties include properties that the Data Integration Service uses when reading data from the source at run time.

The run-time property for Azure Event Hubs event includes the name of the Azure Event Hubs connection.

Advanced Properties

The following table describes the advanced properties for Azure Event Hubs sources:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Consumer Group	The name of the event hub Consumer Group that you read events from.
Max Rate	The maximum number of events that are consumed in a single batch for each partition.
Shared Access Policy Name	The name of the event hub Shared Access Policy. To read from an event hub, you must have Listen permission. If you specify a value for this property, it overwrites the value configured in the Azure Event Hubs connection.
Shared Access Policy Primary Key	The primary key of the event hub Shared Access Policy. If you specify a value for this property, it overwrites the value configured in the Azure Event Hubs connection.
Guaranteed Processing	Ensures that the mapping processes messages that the sources publish and delivers them to the targets at least once. In the event of a failure, there could be potential duplicates but the messages are processed successfully. If the external source or the target is not available, the mapping execution stops to avoid any data loss. By default, the Guaranteed Processing property is selected.
Start Position Offset	The time from which the Azure Event Hubs data object starts reading messages from an event hub. You can select one of the following options: - CUSTOM. Read messages from a specific time. - EARLIEST. Read the earliest messages available on the event hub. - LATEST. Read the latest messages from an event hub.
Custom Start Position Enqueue Time	Required if you set the Start Position Offset property to custom. A UTC timezone datetime value in the ISO8601 format from which the Azure Event Hubs data object starts reading events from an event hub. Specify time in the following format: YYYY-MM-DDThh:mm:ss.sssZ

Column Projections Properties

The following table describes the columns projection properties that you configure for Azure Event Hubs sources:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.

Property	Description
Enable Column Projection	Indicates that you use a schema to read the data that the source streams. By default, the data is streamed in binary format. To change the format in which the data is processed, select this option and specify the schema format.
Schema Format	The format in which the source processes data. You can select one of the following formats: - XML - JSON - Flat - Avro
Use Schema	Specify the XSD schema for the XML format, the sample JSON for the JSON format. Specify a .avsc file for the Avro format.
Use Intelligent Structure Model	Displays the intelligent structure model associated with the complex file. You can select a different model. You can associate the data object with an intelligent structure model and directly parse input from CSV, XML, Avro, or JSON input files. **Note:** - If you disable the column projection, the intelligent structure model associated with the data object is removed. If you want to associate an intelligent structure model with the data object again, enable the column projection and click **Select Model**. - A streaming mapping fails due to schema drift when you use Avro schema with an intelligent structure model. For more information about intelligent structure models, see the *Data Engineering Integration User Guide*.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for sources with hierarchical data. Select this option if the source has hierarchical data. For more information about hierarchical data, see the *Data Engineering Integration User Guide*.

Confluent Kafka Data Objects

A Confluent Kafka data object is a physical data object that represents data in a Kafka stream or a Confluent Kafka stream. After you configure a Messaging connection, create a Confluent Kafka data object to read data from Kafka brokers or from Confluent Kafka brokers using schema registry.

Confluent Kafka runs as a cluster comprised of one or more servers each of which is called a broker. Confluent Kafka brokers stream data in the form of messages. These messages are published to a topic.

Confluent Kafka topics are divided into partitions. The Spark engine can read the partitions of the topics in parallel to achieve better throughput and to scale the number of messages processed. Message ordering is guaranteed only within partitions. For optimal performance use multiple partitions.

You can create or import a Confluent Kafka data object. When you configure a Confluent Kafka data object, you can specify the topic name that you read from. To subscribe to multiple topics that match a pattern, you can specify a regular expression. Before the application runs, the pattern is matched against the topics. If

you add a topic with a similar pattern when the application is already running, the application will not read from the topic.

Read Operation in Confluent Kafka

You can use the Confluent Kafka data object read operation as a source in streaming mappings. By default, the read operation is created for Confluent Kafka.

Supported File Format in Confluent Kafka

When you configure the data operation properties, specify the format in which the Confluent Kafka data object reads data.

You can specify XML, JSON, Avro, or Flat as format for Confluent Kafka data objects. When you specify XML format, you must provide a XSD file. When you specify JSON or Flat format, you must provide a sample file. When you specify Avro format, provide a sample Avro schema in an .avsc file.

You can specify Avro as the format for Confluent Kafka data objects using schema registry. You can pass Avro payload format directly from source to target in streaming mappings using Confluent Kafka.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Note: In Databricks environment, a streaming mapping fails when you enable schema registry in the connection properties of the Confluent Kafka data object.

Confluent Kafka Data Object Overview Properties

You can use the Developer tool to view the overview properties for Confluent Kafka messages in the **Overview** view.

The overview properties include general properties, object properties, and column properties that apply to the Confluent Kafka data object.

General Properties

The following table describes the general properties that you configure for Confluent Kafka data object:

Property	Description
Name	Name of the Confluent Kafka data object.
Description	Description of the Confluent Kafka data object.
Connection	Name of the Confluent Kafka connection.

Objects Properties

The following table describes the objects properties that you configure for Confluent Kafka data object:

Property	Description
Name	Name of the topic or topic pattern of the Confluent Kafka data object.
Description	Description of the Confluent Kafka data object.
Native Name	Native name of Confluent Kafka data object.
Path Information	Type and name of the topic or topic pattern of the Confluent Kafka data object.

Column Properties

The following table describes the column properties that you configure for Confluent Kafka data object:

Property	Description
Name	Name of the Confluent Kafka data object.
Native Name	Native name of the Confluent Kafka data object.
Type	Native data type of the Confluent Kafka data object.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	Scale of the data type.
Description	Description of the Confluent Kafka data object.

Confluent Kafka Header Ports

The Confluent Kafka data object contains default header ports that represent metadata associated with events.

The following table describes the header ports of the Confluent Kafka data object:

Header ports	Data type	Description
partitionId	integer	Partition ID of the data. When the Confluent Kafka topic is divided into partitions, a sequential ID is assigned to each event and is unique to each event within the partition. Use the partition ID to identify the event that you want to consume from a particular partition.
key	binary	Key value associated with an event. You can group events based on the key value and then process the data.
TopicName	string	Name of the Confluent Kafka topic from where you receive events.
timestamp	Date/Time	Time at which an event is generated. You can accumulate streamed data into data groups and then process the data groups based on the timestamp values.

Confluent Kafka Data Object Read Operation Properties

The Data Integration Service uses read operation properties when it reads data from a Confluent Kafka broker.

General Properties

The general properties display the name and description of the Confluent Kafka sources.

The following table describes the general properties that you view for Confluent Kafka sources:

Property	Description
Name	Name of the Confluent Kafka broker read operation. You can edit the name in the **Overview** view. When you use the Confluent Kafka broker as a source in a mapping, you can edit the name in the mapping.
Description	Description of the Confluent Kafka broker read operation.

Ports Properties

The ports properties display the port names and port attributes such as data type and precision of the Confluent Kafka sources.

The following table describes the ports properties that you configure for Confluent Kafka broker sources:

Property	Description
Name	Name of the source header field.
Type	Native data type of the resource.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	Detail of the data type.
Scale	Scale of the data type.
Description	Description of the resource.

Sources Properties

The sources properties display the resources of the Confluent Kafka data object.

The following table describes the sources property that you can configure for Confluent Kafka sources:

Property	Description
Sources	The sources that the Confluent Kafka data object reads from. You can add or remove sources.

Schema Properties

The schema properties display the column related details of the Confluent Kafka sources. To specify schema properties, select the data operation and enable the column projection.

The following table describes the schema properties that you configure for Confluent Kafka sources:

Property	Description
Column Name	Name field that contains data. This property is read-only.
Type	Native data type of the source. This property is read-only.
Enable Column Projection	Indicates that you use a schema to read the data that the source streams.
Schema Format	Format in which the source streams data. Confluent Kafka using schema registry supports Avro format. For Kafka, select one of the following formats: - XML - JSON - Flat - Avro
Use Schema	Specify the XSD schema for the XML format, a sample file for JSON, or .avsc file for Avro format. For the Flat file format, configure the schema to associate a flat file to the Kafka source. When you provide a sample file, the Data Integration Service uses UTF-8 code page when reading the data.
Use Intelligent Structure Model	Displays the intelligent structure model associated with the complex file. You can select a different model. **Note:** - If you disable the column projection, the intelligent structure model associated with the data object is removed. If you want to associate an intelligent structure model with the data object again, enable the column projection and then click **Select Model**. - A streaming mapping with Confluent Kafka source fails validation when you specify a schema registry URL in the Confluent Kafka connection and provide an Avro sample file to Intelligent Structure Discovery. - A streaming mapping fails due to schema drift when you use Avro schema with an intelligent structure model. For more information about intelligent structure models, see the *Data Engineering Integration User Guide*.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.

Run-time Properties

The run-time properties display the name of the connection.

The following table describes the run-time property that you configure for Confluent Kafka sources:

Property	Description
Connection	Name of the Confluent Kafka connection.

Advanced Properties

The advanced properties display the details about operation type, processing, and time from which the Confluent Kafka source starts reading Confluent Kafka messages from a Confluent Kafka topic.

The following table describes the advanced properties that you can configure for Confluent Kafka sources:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Guaranteed Processing	Ensures that the mapping processes the source messages and delivers them to the targets at least once. In the event of a failure, the source messages are duplicated but the messages are processed successfully. If the external source or the target is not available, the mapping execution stops to avoid any data loss. By default, the Guaranteed Processing property is selected.
Start Position Offset	Time that the Confluent Kafka source starts reading messages from a Confluent Kafka topic when you run a mapping. You can select one of the following options: - Custom. Read messages from a specific time. - Earliest. Read the earliest messages available on the Confluent Kafka topic. - Latest. Read messages received by the Confluent Kafka topic after the mapping has been deployed.
Custom Start Position Timestamp	Time in GMT that the Confluent Kafka source starts reading Confluent Kafka messages from a Confluent Kafka topic. Specify a time in the following format: `dd-MM-yyyy HH:mm:ss.SSS` The milliseconds are optional.

Consumer Configuration Properties	Configuration properties for the consumer. If the Confluent Kafka data object is reading data from a Confluent Kafka cluster that is configured for Kerberos authentication, include the following property:
	`security.protocol=SASL_PLAINTEXT,sasl.kerberos.service.name=kafka,`
	`sasl.mechanism=GSSAPI`

Dynamic Schema Strategy	The dynamic schema strategy that identifies the schema at run-time for ingesting streaming data into files or tables in a mapping. Only Avro schema is supported. You must disable the column projection. You can select one of the following strategies: - None - Schema Registry: If you select this, then you can use only Filter, Router, Expression, Java, and Python transformations. Also, the data from KafkaAvroSerializer is supported. - External Location: If you select this, then you can use only Filter, Router, Expression, Java, and Python transformations. Also, the data from ByteArraySerializer is supported. **Note:** Restart the mapping if you want to refresh the schema. Refresh schema is only supported if **Dynamic Schema Strategy** is not selected.
Schema Location	The schema location to fetch the schema in a streaming mapping. If you select `External Location` for the dynamic schema strategy, you must create a `reader.avsc` file having the schema content at the schema location and keep it under the topic name. For example: `<Schema Location>/<Topic Name>/reader.avsc`. Then, specify only the path till the schema location. **Note:** Schema location must be named as per the topic name.

Google PubSub Data Objects

A Google PubSub data object is a physical data object that represents data in a Google PubSub topic. After you create a Google PubSub connection, create a Google PubSub data object to read from Google PubSub. You can use Google PubSub Connector to read Google PubSub topics through subscriptions.

Google PubSub is a messaging service that decouples services that produce events from services that process events. You can use Google PubSub as a messaging-oriented middleware or for event ingestion and delivery for streaming analytics pipelines.

Google PubSub offers durable message storage and real-time message delivery with high availability and consistent performance at scale. You can run Google PubSub servers in all the available Google Cloud regions around the world.

Before you use Google PubSub Connector, you must complete the following prerequisite tasks:

1. Ensure that your organization has the Google PubSub Connector license.

2. Ensure that you have a Google service account to access Google PubSub.

3. Ensure that you have the PROJECTID, CLIENTEMAIL, PRIVATE KEY, PRIVATEKEYID, and CLIENTID values for the Google service account. You will need to enter these details when you create a Google PubSub connection.

4. Ensure that you have enabled the **Google PubSub API** for your Google Cloud project. Google PubSub Connector uses the Google APIs to integrate with Google PubSub.

For more information about the required permissions and the access control options available for a Google PubSub user, see the Google Cloud Pub/Sub documentation.

A Google PubSub topic can have multiple subscriptions. However, one subscription belongs to one single topic. To receive messages published to a topic, you must have a subscription to that topic. Messages are no longer visible in the subscription after they are acknowledged. The Google PubSub connector uses the synchronous pull mechanism to read the subscribed messages.

Note:

-   A streaming mapping with Google PubSub source cannot have multiple pipelines.

-   You cannot enable a streaming mapping with Google PubSub source to override the partition reads of the runtime attributes.

-   You cannot use a Java transformation in a streaming mapping with Google PubSub source.

-   You cannot reduce partition or change Google Pubsub subscription properties during warm start processing.

Google PubSub Data Object Overview Properties

You can use the Developer tool to view the overview properties for Google PubSub messages in the **Overview** view.

The overview properties include general properties, object properties, and column properties that apply to the Google PubSub data object.

General Properties

The following table describes the general properties that you configure for Google PubSub data object:

Property	Description
Name	Name of the Google PubSub data object.
Description	Description of the Google PubSub data object.
Connection	Name of the Google PubSub connection.

Objects Properties

The following table describes the objects properties that you configure for Google PubSub data object:

Property	Description
Name	Name of the topic of the Google PubSub data object.
Description	Description of the Google PubSub data object.
Native Name	Native name of Google PubSub data object.
Path Information	Type and name of the topic of the Google PubSub data object.

Column Properties

The following table describes the column properties that you configure for Google PubSub data object:

Property	Description
Name	Name of the Google PubSub data object.
Native Name	Native name of the Google PubSub data object.
Type	Native data type of the Google PubSub data object.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	Scale of the data type.
Description	Description of the Google PubSub data object.

Google PubSub Data Object Read Properties

The Data Integration Service uses read operation properties when it reads data from a Google PubSub topic.

General Properties

The general properties display the name and description of the Google PubSub sources.

The following table describes the general properties that you view for Google PubSub sources:

Property	Description
Name	Name of the Google PubSub topic read operation. You can edit the name in the **Overview** view. When you use the Google PubSub topic as a source in a mapping, you can edit the name in the mapping.
Description	Description of the Google PubSub topic read operation.

Ports Properties

The ports properties display the port names and port attributes such as data type and precision of the Google PubSub sources.

The following table describes the ports properties that you configure for Google PubSub topic sources:

Property	Description
Name	Name of the source header field.
Type	Native data type of the resource.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	Scale of the data type.
Description	Description of the resource.

Schema Properties

The schema properties display the column related details of the Google PubSub sources. To specify schema properties, select the data operation and enable the column projection.

The following table describes the schema properties that you configure for Google PubSub sources:

Property	Description
Column Name	Name field that contains data. This property is read-only.
Type	Native data type of the source. This property is read-only.
Enable Column Projection	Indicates that you use a schema to read the data that the source streams.
Schema Format	Format in which the source streams data. Select one of the following formats: - Binary - JSON

Property	Description
Schema	Specify the XSD schema for the XML format, a sample file for JSON, or .avsc file for Avro format. When you provide a sample file, the Data Integration Service uses UTF-8 code page when reading the data.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.

Run-time Properties

The run-time properties display the name of the connection.

The following table describes the run-time property that you configure for Google PubSub sources:

Property	Description
Connection	Name of the Google PubSub connection.
Partition Type	Select the partition type. You can select either None or Fixed. To configure fixed partitioning, specify the number of partitions that the Spark Engine creates at runtime. The number of partitions can be equal to the number of CPU cores allocated to the executors. Informatica recommends to increase the number of partitions to increase parallel pull requests. For example, setting `spark.executor.instances` to 10 and `spark.executor.cores` to 2, allocates 20 cores. Set the number of partitions to 20 to create 20 parallel pull requests to read from the Google PubSub source.

Advanced Properties

The advanced properties display the details about operation type, processing, and time from which the Google PubSub source starts reading Google PubSub messages from a Google PubSub topic.

The following table describes the advanced properties that you can configure for Google PubSub sources:

Property	Description
Subscription Starting Pattern	The starting pattern of the subscription name from which the Google PubSub source starts reading messages from a topic when you run a mapping. By default, the subscription name is prefixed with *Infa_<GUID>*.
Acknowledgment Deadline	Amount of time, in seconds, to wait for an acknowledgement for a message that was received by a subscriber, before Google PubSub can retry delivering the same message. Recommended: 150 seconds
Message Retention Duration	Amount of time, in days, to retain unacknowledged messages in the subscription's backlog from the time a message is published. A message that cannot be delivered within the maximum retention time is deleted and is no longer accessible. Default is 7 days.

Property	Description
Subscription Expiration	Time in days after which the subscriptions without any subscriber activity expires. If Google PubSub detects subscriber activity, the subscription deletion clock restarts. Specify 0 for no expiry.
Max Message Count	The maximum number of messages to return for each subscriber request. Google PubSub might return less messages than the specified number of messages. Recommended: 1000 messages
Max Message Size	Maximum incoming message size for each subscriber request, in byte.
Executor Thread Count	Specify the executor thread count for each subscriber request. Default is 4.
Return Immediately	If there are no messages, return immediately for each read request.

HBase Data Objects

An HBase data object is a physical data object that represents data based on an HBase resource. After you create an HBase connection, create an HBase data object and a read data object operation.

Create a data object read operation to when you want to perform an uncached lookup on HBase data.

HBase Data Object Read Operation Properties

HBase data object read operation properties include run-time properties that apply to the HBase data object.

The Developer tool displays advanced properties for the HBase data object operation in the Advanced view.

The following table describes the Advanced properties for an HBase data object read operation:

Property	Description
Operation Type	The type of data object operation. This is a read-only property.
Date Time Format	Format of the columns of the date data type. Specify the date and time formats by using any of the Java date and time pattern strings.
default Column Data Type	Not applicable for streaming mappings.
default Precision	Not applicable for streaming mappings.
default Scale	Not applicable for streaming mappings.
Default Column Family	Not applicable for streaming mappings.
Control File Location	Not applicable for streaming mappings.

JMS Data Objects

A JMS data object is a physical data object that accesses a JMS server. After you configure a JMS connection, create a JMS data object to read from JMS sources.

JMS providers are message-oriented middleware systems that send JMS messages. The JMS data object connects to a JMS provider to read or write data.

The JMS data object can read JMS messages from a JMS provider message queue . When you configure a JMS data object, configure properties to reflect the message structure of the JMS messages. The input ports and output ports are JMS message headers.

When you configure the read data operation properties, specify the format in which the JMS data object reads data. You can specify XML, JSON, or Flat as format. When you specify XML format, you must provide an XSD file. When you specify JSON, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

RELATED TOPICS:

- "Creating a Data Object" on page 139

Integration with JMS

You manually create JMS source and target data objects to reflect the message structure of JMS messages.

The JMS data object can read messages of type TextMessage. This type of message contains a string object. TextMessages can contain XML or JSON message data.

JMS Message Structure

JMS messages contain the following components:

- Header
- Properties
- Body

Header Fields

JMS messages contain a fixed number of header fields. Each JMS message uses these fields regardless of message type. Every JMS source and target definition includes a pre-defined set of header fields.

The following table describes the JMS message header fields:

Header field	Data type	Description
JMSDestination	String	Destination to which the message is sent. JMS destinations can be a message queue or a recipient who listens for messages based on the message topic.
JMSDeliveryMode	Integer	Delivery mode of the message. The delivery mode can be persistent or non-persistent.
JMSMessageID	String	Unique identification value for the message.
JMSTimestamp	Date/Time	Time at which the message was handed off to the provider to be sent to the destination.
JMSCorrelationID	String	Links one message with another. For example, JMSCorrelationID can link a response message with the corresponding request message.
JMSReplyTo	String	Destination to which a reply message can be sent.
JMSRedelivered	String	Indicates that a message might have been delivered previously, but not acknowledged.
JMSType	String	Type of message based on a description of the message. For example, if a message contains a stock trade, the message type might be stock trade.
JMSExpiration	Bigint	Amount of time in milliseconds the message remains valid. The messages remain in memory during this period.
JMSPriority	Integer	Priority of the message from 0-9. 0 is the lowest priority. 9 is the highest.

Property Fields

JMS source and target definitions can optionally include message property fields. Property fields contain additional message header information. JMS providers use properties in a JMS message to give provider-specific information. Applications that use a JMS provider can add property fields with application-specific information to a message.

Body Fields

JMS source and target definitions can optionally include a message body. The body contains one or more fields. Only certain types of JMS messages contain a body.

JMS Data Object Overview Properties

The Data Integration Service uses overview properties when it reads data from a JMS source.

Overview properties include general properties that apply to the JMS data object. They also include object properties that apply to the resources in the JMS data object. The Developer tool displays overview properties for JMS messages in the Overview view.

General Properties

The following table describes the general properties that you configure for JMS data objects:

Property	Description
Name	The name of the JMS data object.
Description	The description of the JMS data object.
Connection	The name of the JMS connection.

Objects Properties

The following table describes the objects properties that you configure for JMS data objects:

Property	Description
Name	The name of the topic or queue of the JMS source.
Description	The description of the JMS source.
Native Name	The native name of JMS source.
Path Information	The type and name of the topic or topic pattern of the JMS source.

JMS Data Object Read Operation Properties

The Data Integration Service uses read operation properties when it reads data from a JMS source. You can edit the format, run-time, and advanced properties.

General Properties

The Developer tool displays general properties for JMS sources in the **Read** view.

The following table describes the general properties that you view for JMS sources:

Property	Description
Name	The name of the JMS read operation. You can edit the name in the **Overview** view. When you use the JMS source as a source in a mapping, you can edit the name in the mapping.
Description	The description of the JMS read operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for JMS sources:

Property	Description
Name	The name of the JMS source header field.
Type	The native data type of the source.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Sources Properties

The sources properties list the resources of the JMS data object. You can add or remove resources in the data object.

Run-time Properties

The run-time properties displays the name of the connection.

The following table describes the run-time property that you configure for JMS sources:

Property	Description
Connection	Name of the JMS connection.

Advanced Properties

The Developer tool displays the advanced properties for JMS sources in the Output transformation in the **Read** view.

You can configure the following advanced properties for JMS sources:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
JMS Destination	Name of the queue or topic to which the JMS provider publishes messages. The data object subscribes to JMS messages from this queue or topic.
Client ID	Client identifier to identify the connection and set up a durable connections.

Property	Description
Durable Subscription Name	Name of the durable subscription that can receive messages sent while the subscribers are not active.
	Durable subscriptions provide the flexibility and reliability of queues, but still allow clients to send messages to many recipients.
	Important: Effective in version 10.4.1.1, you must specify a custom checkpoint directory to retain the generated client ID and durable subscription name suffix. If you do not specify a custom checkpoint directory, the mapping appends a new suffix to the durable subscription name with every run. This action might result in data loss because the mapping does not consume the data sent before creating the new durable subscription name.
JMS Message Selector	Criteria for filtering message header or message properties, to limit which JMS messages the data object receives.
Guaranteed Processing	Ensures that the mapping processes messages that the sources publish and delivers them to the targets at least once. In the event of a failure, there could be potential duplicates but the messages are processed successfully. If the external source or the target is not available, the mapping execution stops to avoid any data loss.
	By default, the Guaranteed Processing property is selected.

Column Projections Properties

The following table describes the columns projection properties that you configure for JMS sources:

Property	Description
Column Name	The name field that contains data.
	This property is read-only.
Type	The native data type of the resource.
	This property is read-only.
Enable Column Projection	Indicates that you use a schema to read the data that the source streams. By default, the data is streamed in binary format. To change the format in which the data is streamed, select this option and specify the schema format.
Schema Format	The format in which the source streams data. You can select one of the following formats:
	- XML
	- JSON
	- Flat
Use Schema	Specify the XSD schema for the XML format or the sample JSON for the JSON format.
	For the Flat file format, configure the schema to associate a flat file to the source. When you provide a sample file, the Data Integration Service uses UTF-8 code page when reading the data.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for hierarchical data.
	For more information, see the *Data Engineering Integration User Guide*.

Configuring a Schema for Flat Files

Configure schema for flat files when you configure column projection properties.

1. On the **Column Projection** tab, enable column projection and select the flat schema format.

    The page displays the column projection properties page.

2. On the column projection properties page, configure the following properties:

    - Sample Metadata File. Select a sample file.

    - Code page. Select the UTF-8 code page.

    - Format. Format in which the source processes data. Default value is Delimited. You cannot change it.

3. Click **Next**.

4. In the delimited format properties page, configure the following properties:

Property	Description
Delimiters	Specify the character that separates entries in the file. Default is a comma (,). You can only specify one delimiter at a time. If you select **Other** and specify a custom delimiter, you can only specify a single-character delimiter.
Text Qualifier	Specify the character used to enclose text that should be treated as one entry. Use a text qualifier to disregard the delimiter character within text. Default is No quotes. You can only specify an escape character of one character.
Preview Options	Specify the escape character. The row delimiter is not applicable as only one row is created at a time.
Maximum rows to preview	Specify the rows of data you want to preview.

5. Click **Next** to preview the flat file data object.

    If required, you can change the column attributes. The data type timestampWithTZ format is not supported.

6. Click **Finish**.

    The data object opens in the editor.

Kafka Data Objects

A Kafka data object is a physical data object that represents data in a Kafka stream. After you configure a Messaging connection, create a Kafka data object to read from Apache Kafka brokers.

Kafka runs as a cluster comprised of one or more servers each of which is called a broker. Kafka brokers stream data in the form of messages. These messages are published to a topic.

Kafka topics are divided into partitions. Spark Structured Streaming can read the partitions of the topics in parallel. This gives better throughput and could be used to scale the number of messages processed. Message ordering is guaranteed only within partitions. For optimal performance you should have multiple partitions. You can create or import a Kafka data object.

When you configure the Kafka data object, specify the topic name that you read from or write to. You can specify the topic name or use a regular expression for the topic name pattern only when you read from Kafka. To subscribe to multiple topics that match a pattern, you can specify a regular expression. When you run the application on the cluster, the pattern matching is done against topics before the application runs. If you add a topic with a similar pattern when the application is already running, the application will not read from the topic.

After you create a Kafka data object, create a read operation. You can use the Kafka data object read operation as a source in streaming mappings. If you want to configure high availability for the mapping, ensure that the Kafka cluster is highly available. You can also read from a Kerberised Kafka cluster.

When you configure the data operation read properties, you can specify the time from which the Kafka source starts reading Kafka messages from a Kafka topic.

You can associate the data object with an intelligent structure model and directly parse input from text, CSV, XML, Avro, or JSON input files.

When you configure the data operation properties, specify the format in which the Kafka data object reads data. You can specify XML, JSON, Avro, or Flat as format. When you specify XML format, you must provide a XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON or Flat format, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

For more information about Kafka clusters, Kafka brokers, and partitions, see
http://kafka.apache.org/11/documentation.html.

For more information about how to use topic patterns in Kafka data objects, see
https://docs.informatica.com/data-engineering/data-engineering-streaming/h2l/1132-how-to-use-topic-patterns-in-kafka-data-objects/abstract.html.

Note: If you use a Kafka data object in a streaming mapping, you cannot use a MapR Streams data object in the same mapping.

Kafka Combinations Support Matrix

The following table shows the different authentication mechanisms that you can use against listener protocols in a streaming mapping with a Kafka source:

Protocols	Authentication Mechanisms		
	Plain	**GSSAPI**	**None**
SSL+TLS1.0	Yes	Yes	Yes
SSL+TLS1.1	Yes	Yes	Yes
SSL+TLS1.2	Yes	Yes	Yes
SSL+TLS1.0,TLS1.1,TLS1.2	Yes	Yes	Yes

Protocols	Authentication Mechanisms		
	Plain	GSSAPI	None
SASL_SSL+TLS1.0	Yes	Yes	NA
SASL_SSL+TLS1.1	Yes	Yes	NA
SASL_SSL+TLS1.2	Yes	Yes	NA
SASL_SSL+TLS1.0,TLS1.1,TLS1.2	Yes	Yes	NA

RELATED TOPICS:

-

Kafka Data Object Overview Properties

The Data Integration Service uses overview properties when it reads data from or writes data to a Kafka broker.

Overview properties include general properties that apply to the Kafka data object. They also include object properties that apply to the resources in the Kafka data object. The Developer tool displays overview properties for Kafka messages in the **Overview** view.

General Properties

The following table describes the general properties that you configure for Kafka data objects:

Property	Description
Name	The name of the Kafka read operation.
Description	The description of the Kafka data object.
Connection	The name of the Kafka connection.

Objects Properties

The following table describes the objects properties that you configure for Kafka data objects:

Property	Description
Name	The name of the topic or topic pattern of the Kafka data object.
Description	The description of the Kafka data object.
Native Name	The native name of Kafka data object.
Path Information	The type and name of the topic or topic pattern of the Kafka data object.

Column Properties

The following table describes the column properties that you configure for Kafka data objects:

Property	Description
Name	The name of the Kafka data object.
Native Name	The native name of the Kafka data object.
Type	The native data type of the Kafka data object.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	The scale of the data type.
Description	The description of the Kafka data object.
Access Type	The type of access the port or column has.

Kafka Header Ports

The Kafka data object contains default header ports that represent metadata associated with events.

The following table lists the header ports of the Kafka data object:

Header ports	Data type	Description
partitionId	integer	Partition ID of the data. When the Kafka topic is divided into partitions, a sequential ID is assigned to each event and is unique to each event within the partition. Use the partition ID to identify the event that you want to consume from a particular partition.
key	binary	Key value associated with an event. You can group events based on the key value and then process the data.
TopicName	string	Name of the Kafka topic from where you receive events.
timestamp	Date/Time	Time at which an event is generated. You can accumulate streamed data into data groups and then process the data groups based on the timestamp values.

Kafka Data Object Read Operation Properties

The Data Integration Service uses read operation properties when it reads data from a Kafka broker.

General Properties

The Developer tool displays general properties for Kafka sources in the **Read** view.

The following table describes the general properties that you view for Kafka sources:

Property	Description
Name	The name of the Kafka broker read operation. You can edit the name in the **Overview** view. When you use the Kafka broker as a source in a mapping, you can edit the name in the mapping.
Description	The description of the Kafka broker read operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Kafka broker sources:

Property	Description
Name	The name of the source header field.
Type	The native data type of the resource.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Run-time Properties

The run-time properties displays the name of the connection.

The following table describes the run-time property that you configure for Kafka sources:

Property	Description
Connection	Name of the Kafka connection.

Advanced Properties

The Developer tool displays the advanced properties for Kafka sources in the Output transformation in the Read view.

The following table describes the advanced properties that you can configure for Kafka sources:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Guaranteed Processing	Ensures that the mapping processes messages that the sources publish and delivers them to the targets at least once. In the event of a failure, there could be potential duplicates but the messages are processed successfully. If the external source or the target is not available, the mapping execution stops to avoid any data loss. By default, the Guaranteed Processing property is selected.
Start Position Offset	The time from which the Kafka source starts reading messages from a Kafka topic when you run a mapping. You can select one of the following options: - Custom. Read messages from a specific time. - Earliest. Read the earliest messages available on the Kafka topic. - Latest. Read messages received by the Kafka topic after the mapping has been deployed.
Custom Start Position Timestamp	The time in GMT from which the Kafka source starts reading Kafka messages from a Kafka topic. Specify a time in the following format: `dd-MM-yyyy HH:mm:ss.SSS` The milliseconds are optional.
Consumer Configuration Properties	The configuration properties for the consumer. If the Kafka data object is reading data from a Kafka cluster that is configured for Kerberos authentication, include the following property: `security.protocol=SASL_PLAINTEXT,sasl.kerberos.service.name=kafka, sasl.mechanism=GSSAPI`
Dynamic Schema Strategy	The dynamic schema strategy that identifies the schema at run-time for ingesting streaming data into files or tables in a mapping. Only Avro schema is supported. You must disable the column projection. You can select one of the following strategies: - None - External Location: If you select this, then you can use only Filter, Router, Expression, Java, and Python transformations. Also, the data from ByteArraySerializer is supported. **Note:** Restart the mapping if you want to refresh the schema. Refresh schema is only supported if **Dynamic Schema Strategy** is not selected.
Schema Location	The schema location to fetch the schema in a streaming mapping. If you select `External Location` for the dynamic schema strategy, you must create a `reader.avsc` file having the schema content at the schema location and keep it under the topic name. For example: `<Schema Location>/<Topic Name>/reader.avsc`. Then, specify only the path till the schema location. **Note:** Schema location must be named as per the topic name.

Sources Properties

The sources properties list the resources of the Kafka data object.

The following table describes the sources property that you can configure for Kafka sources:

Property	Description
Sources	The sources which the Kafka data object reads from. You can add or remove sources.

Column Projection Properties

The Developer tool displays the column projection properties in the Properties view of the Read operation.

To specify column projection properties, double click on the read operation and select the data object. The following table describes the columns projection properties that you configure for Kafka sources:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the source. This property is read-only.
Enable Column Projection	Indicates that you use a schema to read the data that the source streams. By default, the data is streamed in binary format. To change the format in which the data is streamed, select this option and specify the schema format.
Schema Format	The format in which the source streams data. Select one of the following formats: - XML - JSON - Flat - Avro
Use Schema	Specify the XSD schema for the XML format, a sample file for JSON, or .avsc file for Avro format. For the Flat file format, configure the schema to associate a flat file to the Kafka source. When you provide a sample file, the Data Integration Service uses UTF-8 code page when reading the data.
Use Intelligent Structure Model	Displays the intelligent structure model associated with the complex file. You can select a different model. You can associate the data object with an intelligent structure model and directly parse input from CSV, XML, Avro, or JSON input files. **Note:** - If you disable the column projection, the intelligent structure model associated with the data object is removed. If you want to associate an intelligent structure model with the data object again, enable the column projection and click **Select Model**. - A streaming mapping fails due to schema drift when you use Avro schema with an intelligent structure model. For more information about intelligent structure models, see the *Data Engineering Integration User Guide*.

Property	Description
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for hierarchical data. For more information about hierarchical data, see the *Data Engineering Integration User Guide*.

Configuring Schema for Flat Files

Configure schema for flat files when you configure column projection properties.

1. On the **Column Projection** tab, enable column projection and select the flat schema format.

    The page displays the column projection properties page.

2. On the column projection properties page, configure the following properties:

    - Sample Metadata File. Select a sample file.

    - Code page. Select the UTF-8 code page.

    - Format. Format in which the source processes data. Default value is Delimited. You cannot change it.

3. Click **Next**.

4. In the delimited format properties page, configure the following properties:

Property	Description
Delimiters	Specify the character that separates entries in the file. Default is a comma (,). You can only specify one delimiter at a time. If you select **Other** and specify a custom delimiter, you can only specify a single-character delimiter.
Text Qualifier	Specify the character used to enclose text that should be treated as one entry. Use a text qualifier to disregard the delimiter character within text. Default is No quotes. You can only specify an escape character of one character.
Preview Options	Specify the escape character. The row delimiter is not applicable as only one row is created at a time.
Maximum rows to preview	Specify the rows of data you want to preview.

5. Click **Next** to preview the flat file data object.

    If required, you can change the column attributes. The data type timestampWithTZ format is not supported.

6. Click **Finish**.

    The data object opens in the editor.

MapR Streams Data Objects

A MapR Streams data object is a physical data object that represents data in a MapR Stream. After you create a MapR Streams connection, create a MapR Streams data object to read data from MapR Streams.

Before you create and use MapR Stream data objects in streaming mappings, complete the required prerequisites.

For more information about the prerequisite tasks, see the *Data Engineering Integration Guide*.

When you configure the MapR Streams data object, specify the stream name that you read from in the following format:

```
/pathname:topic name
```

You can specify the stream name or use a regular expression for the stream name pattern only when you read from MapR Streams. The regular expression that you specify applies to the topic name and not the path name. To subscribe to multiple topics that match a pattern, you can specify a regular expression. When you run the application on the cluster, the pattern matching is done against topics before the application runs. If you add a topic with a similar pattern when the application is already running, the application will not read from the topic.

After you create a MapR Streams data object, create a read data object operation. You can then add the data object read operation as a source in streaming mappings.

You can associate the data object with an intelligent structure model and directly parse input from text, CSV, XML, or JSON input files.

When you configure the data operation properties, specify the format in which the MapR Streams data object reads data. You can specify XML, JSON, or Avro as format. When you specify XML format, you must provide a XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON or Flat format, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

For more information about how to use topic patterns in MapR Streams data objects, see https://docs.informatica.com/data-engineering/data-engineering-streaming/h2l/1149-how-to-use-topic-patterns-in-maprstreams-data-objects/abstract.html.

Note: If you use a MapR Streams data object in a streaming mapping, you cannot use an Apache Kafka data object in the same mapping.

MapR Streams Object Overview Properties

Overview properties include general properties that apply to the MapR Streams data object. The Developer tool displays overview properties in the Overview view.

General

Property	Description
Name	Name of the MapR Streams data object.
Description	Description of the MapR Streams data object.
Native Name	Native name of the MapR Stream.
Path Information	The path of the MapR Stream.

Column Properties

The following table describes the column properties that you configure for MapR Streams data objects:

Property	Description
Name	The name of the MapR Streams data object.
Native Name	The native name of the MapR Streams data object.
Type	The native data type of the MapR Streams data object.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	The scale of the data type.
Description	The description of the MapR Streams data object.
Access Type	The access type of the port or column.

MapR Streams Header Ports for a Source

The MapR Streams data object contains default header ports that represent metadata associated with events.

The following table lists the header ports of the MapR Streams data object when you use it as a source:

Header ports	Data type	Description
partitionId	integer	Partition ID of the data. When a MapR stream is divided into partitions, a sequential ID is assigned to each event and is unique to each event within the partition. Use the partition ID to identify the event that you want to consume from a particular partition.
key	binary	Key value associated with an event. You can group events based on the key value and then process the data.
TopicName	string	Name of the stream from which you receive events.
timestamp	date/time	Time at which an event is generated. You can accumulate streamed data into data groups and then process the data groups based on the timestamp values.

MapR Streams Data Object Read Operation Properties

The Data Integration Service uses read operation properties when it reads data from MapR Streams.

General Properties

The Developer tool displays general properties for MapR Streams sources in the **Read** view.

The following table describes the general properties for the MapR Streams data object read operation:

Property	Description
Name	The name of the MapR Streams read operation. You can edit the name in the **Overview** view. When you use the MapR Streams as a source in a mapping, you can edit the name in the mapping.
Description	The description of the MapR Streams read operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for MapR Stream sources:

Property	Description
Name	The name of the source header field.
Type	The native data type of the resource.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.

Property	Description
Scale	The scale of the data type.
Description	The description of the resource.

Sources Properties

The sources properties list the resources of the MapR Streams data object.

The following table describes the sources property that you can configure for MapR Stream sources:

Property	Description
Sources	The sources which the MapR Streams data object reads from. You can add or remove sources.

Run-time Properties

The run-time property for MapR Stream source includes the name of the MapR Stream connection.

Advanced Properties

The Developer tool displays the advanced properties for MapR Stream sources in the Output transformation in the **Read** view.

The following table describes the advanced properties for MapR Stream sources:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Guaranteed Processing	Ensures that the mapping processes messages that the sources publish and delivers them to the targets at least once. In the event of a failure, there could be potential duplicates but the messages are processed successfully. If the external source or the target is not available, the mapping execution stops to avoid any data loss. Select this option to avoid data loss.

Column Projections Properties

The following table describes the columns projection properties that you configure for MapR Stream sources:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.

Property	Description
Enable Column Projection	Indicates that you use a schema to read the data that the source streams. By default, the data is streamed in binary format. To change the format in which the data is streamed, select this option and specify the schema format.
Schema Format	The format in which the source streams data. You can select one of the following formats: - XML - JSON - Avro
Add Schema	Specify the XSD schema for the XML format, a sample JSON for the JSON format, or .avsc file for the Avro format.
Use Intelligent Structure Model	Displays the intelligent structure model associated with the complex file. You can select a different model. **Note:** If you disable the column projection, the intelligent structure model associated with the data object is removed. If you want to associate an intelligent structure model with the data object again, enable the column projection and click **Select Model**. For more information about intelligent structure models, see the *Data Engineering Integration User Guide*.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for hierarchical data. For more information about hierarchical, see the *Data Engineering Integration User Guide*.

CHAPTER 4

Targets in a Streaming Mapping

This chapter includes the following topics:

Targets in a Streaming Mapping Overview

You can access log file data, sensor data, Supervisory Control And Data Acquisition (SCADA) data, message bus data, Programmable logic controller (PLC) data on the Spark SQL engine in the Hadoop environment or Databricks Spark SQL engine in the Databricks environment.

You can create physical data objects to access the different types of data. Based on the type of target you are writing to, you can create the relevant data objects.

Targets in a Streaming Mapping on Hadoop and Databricks

A streaming mapping that runs in the Hadoop environment can include file, database, and streaming targets. A streaming mapping that runs in the Databricks environment can include file and streaming targets.

The following table lists the data objects that you can create based on the type of target you write to in either Hadoop or Databricks environment:

Targets in a Streaming Mapping	Environment
Amazon Kinesis	Hadoop, Databricks
Amazon S3	Hadoop, Databricks
Azure Event Hubs	Hadoop, Databricks
Cassandra	Hadoop
Complex File	Hadoop
Confluent Kafka	Databricks
Databricks Delta Lake	Databricks
Google Cloud Storage	Hadoop
HBase	Hadoop
JMS	Hadoop
Kafka	Hadoop, Databricks
Kudu	Hadoop
MapR Streams	Hadoop
Microsoft Azure Data Lake	Hadoop
Microsoft Azure Data Lake Storage Gen2	Hadoop, Databricks
Relational	Hadoop
Snowflake	Hadoop

Note: In a Databricks environment, you cannot use CSV and Parquet payloads in a streaming mapping with Amazon S3 targets.

You can run streaming mappings in the Databricks environment on the AWS or Azure platforms. The following table shows the list of targets that you can include in a streaming mapping based on the cloud platform:

Targets	Cloud Platform
Amazon Kinesis	AWS
Amazon S3	AWS
Azure Event Hubs	Azure
Confluent Kafka	AWS, Azure
Databricks Delta Lake	AWS, Azure
Kafka	AWS, Azure
Microsoft Azure Data Lake Storage Gen2	Azure

Processing Hierarchical Data in Streaming Mappings

Data objects in a streaming mapping can process hierarchical data through complex data types. If you want to project data as complex data types, you must enable the write data operation to project data as complex data types.

The following table shows the format and complex data types that targets in a streaming mapping support:

Object Type	File Type	Supported Formats
Amazon S3	Avro JSON Parquet	Hierarchical Flat
Azure Data Lake Storage Gen1	Avro Binary JSON XML	Flat Hierarchical Binary
Azure Data Lake Storage Gen2	Avro JSON Parquet	Flat Hierarchical Binary
Azure Event Hub	Avro Binary JSON XML	Flat Hierarchical Binary

Object Type	File Type	Supported Formats
Complex File	Avro Binary JSON XML Parquet	Flat Hierarchical Binary
Confluent Kafka	Avro Binary Flat JSON XML	Flat Hierarchical Binary
Databricks Delta Lake	Avro Binary JSON XML	Flat Binary
Kafka	Avro Binary Flat JSON XML	Flat Hierarchical Binary
MapR Streams	Avro Binary JSON XML	Flat Hierarchical Binary

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Amazon Kinesis Data Objects

An Amazon Kinesis data object is a physical data object that represents data in an Amazon Kinesis Data Firehose Delivery Stream. After you create an Amazon Kinesis connection, create an Amazon Kinesis data object to write to Amazon Kinesis Data Firehose.

Kinesis Data Firehose is a real-time data stream processing option that Amazon Kinesis offers within the AWS ecosystem. Kinesis Data Firehose allows batching, encrypting, and compressing of data. Kinesis Data Firehose can automatically scale to meet system needs.

When you configure the Amazon Kinesis data object, specify the name of the Data Firehose Delivery Stream that you write to. You can specify the Kinesis Data Firehose Delivery Stream name or use a regular expression for the stream name pattern. If the input has multiple partitions, you can create multiple Kinesis Data Firehose Delivery Streams to the same target and send the data from these partitions to the individual delivery streams based on the pattern you specify in the stream name.

After you create the data object, create a data object write operation to write data to an Amazon Kinesis Data Firehose Delivery Stream. You can then add the data object write operation as a target in Streaming mappings.

When you configure the data operation properties, specify the format in which the data object writes data. When you write to Amazon Data Firehose targets, you can specify JSON or binary as the format.

When you specify JSON format, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

When you run a mapping to write data to an Amazon Kinesis Data Firehose Delivery Stream, the data object uses the AWS Firehose SDK to write data.

Note: You cannot run a mapping with an Amazon Kinesis data object on MapR and Azure HDInsight distributions.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

For more information about Kinesis Data Firehose, see the Amazon Web Services documentation.

Amazon Kinesis Data Object Overview Properties

Overview properties include general properties that apply to the Amazon Kinesis data object. The Developer tool displays overview properties of the data object in the **Overview** view.

You can configure the following overview properties for Amazon Kinesis data objects:

General

> You can configure the following general properties for the Amazon Kinesis data object:

- Name. Name of the Amazon Kinesis operation.
- Description. Description of the Amazon Kinesis data object.
- Native Name. Name of the Amazon Kinesis data object.
- Path Information. The path of the data object in Amazon Kinesis. For example, `/DeliveryStreams/router1`

Column

> You can configure the name, native name, data type, precision, access type, scale, and description of the columns in the Amazon Kinesis resource.

Advanced

> The following are the advanced properties for the Amazon Kinesis data object:

- Amazon Resource Name. The Kinesis resource that the Amazon Kinesis data object is reading from or writing to.
- Type. The type of delivery stream that the Amazon Kinesis data object is reading from or writing to. The delivery stream is either Kinesis Stream or Firehose DeliveryStream
- Number of Shards. Specify the number of shards that the Kinesis Stream is composed of. This property is not applicable for Firehose DeliveryStream.

Amazon Kinesis Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to Amazon Kinesis Firehose.

General Properties

The Developer tool displays general properties for Amazon Kinesis targets in the **Write** view.

The following table describes the general properties that you view for Amazon Kinesis targets:

Property	Description
Name	The name of the Amazon Kinesis target write operation. You can edit the name in the **Overview** view. When you use the Amazon S3 target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Amazon Kinesis target write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Amazon Kinesis targets:

Property	Description
Name	The name of the target.
Type	The native data type of the target.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the target.

Target Properties

The targets properties list the targets of the Amazon Kinesis data object.

The following table describes the sources property that you can configure for Kinesis Firehose targets:

Property	Description
Target	The target which the Amazon Kinesis data object writes to. You can add or remove targets.

Run-time Properties

The run-time properties include properties that the Data Integration Service uses when writing data to the target at run time, such as reject file names and directories.

The run-time property for Amazon Kinesis targets includes the name of the Amazon Kinesis connection.

Advanced Properties

Advanced properties include tracing level, row order, and retry attempt properties.

The following table describes the advanced properties for Amazon Kinesis targets:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Record Delimiter	The record delimiter that is inserted into the Kinesis Firehose delivery stream.
Maximum Error Retry Attempts	The number of times that the Data Integration Service attempts to reconnect to the target.
Response Wait Time (milliseconds)	The number of milliseconds that the Data Integration Service waits for a response to send a batch request.
Retry Attempt Delay Time (milliseconds)	The number of milliseconds that the Data Integration Service waits before it retries to send data to the Kinesis Firehose delivery stream.
Runtime Properties	The runtime properties for the connection pool and AWS client configuration. Specify the properties as key -value pairs. **For example:** `key1=value1,key2=value2`

Column Projection Properties

The following table describes the columns projection properties that you configure for Amazon Kinesis Firehose targets:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.
Enable Column Projection	Indicates that you use a schema to write data to the target. By default, the data is streamed in binary format. To change the format in which the data is streamed, select this option and specify the schema format.
Schema Format	The format in which data is written to the target. You can select one of the following formats: - JSON - Binary

Property	Description
Use Schema	Specify a sample file for JSON.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for hierarchical data. For more information on hierarchical data, see the *Data Engineering Integration User Guide*.

Amazon S3 Data Objects

An Amazon S3 data object is a physical data object that represents data in an Amazon S3 resource. After you configure an Amazon S3 connection, create an Amazon S3 data object to write to Amazon S3 targets.

You can configure the data object write operation properties that determine how data can be loaded to Amazon S3 targets. After you create an Amazon S3 data object, create a write operation. You can use the Amazon S3 data object write operation as a target in streaming mappings. You can create the data object write operation for the Amazon S3 data object automatically. Then, edit the advanced properties of the data object write operation and run a mapping.

When you configure the data operation properties, specify the format in which the data object writes data. You can specify Avro, Parquet, or JSON as format. When you specify Avro format, provide a sample Avro schema in an `.avsc` file. When you specify JSON, you must provide a sample file.

You can pass any payload format directly from source to target in streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Note: You cannot use binary and CSV payloads in a streaming mapping with Amazon S3 targets.

To successfully run a streaming mapping when you select multiple objects from different Amazon S3 buckets, ensure that all the Amazon S3 buckets belong to the same region and use the same credentials to access the Amazon S3 buckets.

Note: You cannot run a mapping with an Amazon S3 data object on MapR and Azure HDInsight distributions.

RELATED TOPICS:

-

Amazon S3 Data Object Properties

The Amazon S3 Overview view displays general information about the Amazon S3 data object and the object properties that apply to the Amazon S3 files you import. The Developer tool displays overview properties for Amazon S3 in the **Overview** view.

General Properties

You can configure the following properties for an Amazon S3 data object:

- Name. Name of the Amazon S3 data object.
- Description. Description of the Amazon S3 data object.

- Native Name. Name of the Amazon S3 data object.

- Path Information. The path of the data object in Amazon S3. For example, `/s3.demo/test1/`

Column

You can configure the name, native name, data type, precision, access type, scale, and description of the columns in the Amazon S3 data object.

Amazon S3 Data Object Write Operation Properties

The Data Integration Service writes data to an Amazon S3 object based on the data object write operation. The Developer tool displays the data object write operation properties for the Amazon S3 data object in the **Data Object Operation** section.

You can edit the format, run-time, and advanced properties.

General Properties

The Developer tool displays general properties for Amazon S3 targets in the **Write** view.

The following table describes the general properties that you view for Amazon S3 targets:

Property	Description
Name	The name of the Amazon S3 target write operation. You can edit the name in the **Overview** view. When you use the Amazon S3 target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Amazon S3 target write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Amazon S3 targets:

Property	Description
Name	The name of the port.
Type	The data type of the port.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the port.

Target Properties

The target properties list the resources of the Amazon S3 data object. You can add or remove resources in the data object.

Run-time Properties

The run-time properties display the name of the connection used for write transformation.

The following table describes the run-time properties that you configure for an Amazon S3 write operation:

Property	Description
Connection	Name of the Amazon S3 connection.
Partition Type	You can configure multiple partitions when you write to an Amazon S3 target. You can either select **None** or **Dynamic** partition type.

Advanced Properties

The Developer tool displays the advanced properties for Amazon S3 targets in the Input transformation in the **Write** view.

You can configure the following advanced properties for Amazon S3 targets:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Overwrite File(s) If Exists	Not applicable for streaming mappings.
Folder Path	Bucket name or folder path of the Amazon S3 source file that you want to write to. If applicable, include the folder name that contains the target file in the `<bucket_name>`/`<folder_name>` format. If you do not provide the bucket name and specify the folder path starting with a slash (/) in the `/<folder_name>` format, the folder path appends with the folder path that you specified in the connection properties. For example, if you specify the `<my_bucket1>`/`<dir1>` folder path in the connection property and `/<dir2>` folder path in this property, the folder path appends with the folder path that you specified in the connection properties in `<my_bucket1>`/`<dir1>`/`<dir2>` format. If you specify the `<my_bucket1>`/`<dir1>` folder path in the connection property and `<my_bucket2>`/`<dir2>` folder path in this property, the Data Integration Service writes the file in the `<my_bucket2>`/`<dir2>` folder path that you specify in this property.
File Name	Name of the Amazon S3 target file that you want to write to. The file specified at run time overrides the file specified while creating a connection.

Property	Description
Encryption Type	Method you want to use to encrypt data. Select one of the following values: - None - Server Side Encryption Client side encryption and server side encryption with KMS are not applicable for streaming mappings.
Staging Directory	Not applicable for streaming mappings.
File Merge	Merges the target files into a single file. Not applicable for streaming mappings.
Hadoop Performance Tuning Options	Not applicable for streaming mappings.
Compression Format	Compresses data when you write data to Amazon S3. You can compress the data in the following formats: - **None** - **Bzip2** - **Deflate** - **Gzip** - **Lzo** - **Snappy** - **Zlib** Default is **None**.
Object Tags	Not applicable for streaming mappings.
TransferManager Thread Pool Size	The number of threads to write data in parallel. Default is 10. When the file size is more than 5 MB, you can configure multipart upload to upload object in multiple parts in parallel. If you set the value of the **TransferManager Thread Pool Size** to greater than 50, the value reverts to 50. Not applicable for streaming mappings.
Part Size	The part size in bytes to upload an Amazon S3 object. Default is 5 MB. Not applicable for streaming mappings.
Temporary Credential Duration	The time duration during which an IAM user can use the dynamically generated temporarily credentials to access the AWS resource. Enter the time duration in seconds. Default is 900 seconds. If you require more than 900 seconds, you can set the time duration maximum up to 12 hours in the AWS console and then enter the same time duration in this property.
Stream Rollover Size in GB	Optional. Target file size, in gigabytes (GB), at which to trigger rollover. A value of zero (0) means that the target file does not roll over based on size. Default is 1 GB.
Stream Rollover Time in Hours	Optional. Length of time, in hours, for a target file to roll over. After the time period has elapsed, the target file rolls over. A value of zero (0) means that the target file does not roll over based on time. Default is 1 hour.
Interim Directory	The active directory location of the complex file target. This directory stages all the files currently in open state. When the stream rollover condition is met, the files are moved from the interim directory to the target directory.

Column Projection Properties

You can use the advanced properties to specify data object write operation properties to write data to Amazon S3.

The following table describes the columns projection properties that you configure for Amazon S3 targets:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.
Enable Column Projection	Uses a schema to publish the data to the target. Amazon S3 does not support binary format. To change the format in which the data is streamed, select this option and specify the schema format. Default is disabled.
Schema Format	The format in which you stream data to the target. You can select one of the following formats: - Avro - JSON - Parquet
Use Schema	Specify the sample JSON for the JSON format.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.

FileName Port in Amazon S3

When you create a data object read or write operation for Amazon S3 files, the FileName port appears by default.

When the Spark engine writes to Amazon S3 files using a FileName port, it uses the following process to write the data:

1. The Data Integration Service creates separate directories for each value in the FileName port and adds the target files within the directories.

2. The file rollover process closes the current file to which data is being written to and creates a new file based on the configured rollover value.
   Effective in 10.4.1, you can configure the following optional execution rollover parameters at the design-time based on time and size:

   - Stream Rollover Time in Hours. Specify the rollover time in hours for a target file when a certain period of time has elapsed.

   - Stream Rollover Size in GB. Specify the size in GB for a target file when the target file reaches a certain size.

3. When a target file reaches the configured rollover value, the Spark engine rolls over and moves the target file to the specified Amazon S3 target location.

4. The Spark engine creates sub-directories in the specified Amazon S3 target location for each value in the FileName port.

5. The Spark engine moves the rolled over target files to the sub-directories created for each value in the FileName port in the specified Amazon S3 target location.

By default, the Spark engine rolls over based on file size. You can configure both rollover schemes for an Amazon S3 target file. The Spark engine rolls over to based on the first event that triggers. For example, if you configure rollover time to 1 hour and rollover size to 1 GB, the target service rolls the file over when the file reaches a size of 1 GB even if the 1 hour period has not elapsed.

For more information about FileName port, see the *PowerExchange for Amazon S3 User Guide*.

Azure Event Hubs Data Objects

An Azure Event Hubs data object is a physical data object that represents data in Microsoft Azure Event Hubs data streaming platform and event ingestion service. After you create an Azure Event Hubs connection, create an Azure Event Hubs data object to write to an event hub event.

Azure Event Hubs is a highly scalable data streaming platform and event ingestion service, that receives and processes events. Azure Event Hubs can process and store events or data produced by distributed software and devices.

When you configure the Azure Event Hubs data object, specify the name of the event that you write to. After you create the data object, create a data object write operation to write data to an event hub. You can then add the data object write operation as a target in streaming mappings.

Configure partition keys to separate events into different partitions. Events with the same partition key are sent to the same partition on the event hub. Events that are sent in a batch do not need to have the same partition key.

When you configure the data operation properties, specify the format in which the Azure Event Hubs data object writes data. You can specify XML, JSON, Avro, or Flat as format. When you specify XML format, you must provide a XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON or Flat format, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Azure Event Hubs Data Object Overview Properties

Overview properties include general properties that apply to the Azure Event Hubs data object. The Developer tool displays overview properties of the data object in the **Overview** view.

You can configure the following overview properties for Azure Event Hubs data objects:

General

> You can configure the following general properties for the Azure Event Hubs data object:

- Name. Name of the Azure Event Hubs operation.
- Description. Description of the Azure Event Hubs data object.
- Native Name. Name of the Azure Event Hubs data object.

- Path Information. The path of the data object in Azure Event Hubs. For example, `/EventHubs/avroevents`

Column

You can configure the name, native name, data type, precision, access type, scale, and description of the columns in the Azure Event Hubs resource.

Advanced

The following are the advanced properties for the Azure Event Hubs data object:

- Location. The location of the Azure Event Hubs.
- Date of Creation. The date of creation of the Azure Event Hubs.
- Partition Count. The number of partitions that the Event Hub has when you import the data object.

Azure Event Hubs Header Ports

An Azure Event Hubs data object contains default header port that represent metadata associated with events.

An Azure Event Hubs data object contains the following header ports:

partitionKey

Partition keys to separate events into different partitions. Events with same partition key are sent to the same partition on the event hub.

timestamp

Time at which an event is generated. You can accumulate streamed data into data groups and then process the data groups based on the timestamp values.

Azure Event Hubs Data Object Write Operations

The Data Integration Service uses write operation properties when it writes data to an Azure Event Hubs.

General Properties

The Developer tool displays general properties for Azure Event Hub sources in the **Write** view.

The following table describes the general properties for the Azure Event Hub data object write operation:

Property	Description
Name	The name of the Azure Event Hub data object write operation. You can edit the name in the **Overview** view. When you use the Azure Event Hub as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Azure Event Hub data object write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Azure Event Hub targets:

Property	Description
Name	The name of the target.
Type	The native data type of the target.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Target Properties

The targets properties list the targets of the Azure Event Hub data object.

The following table describes the sources property that you can configure for Azure Event Hub targets:

Property	Description
Target	The target which the Azure Event Hub data object writes to. You can add or remove targets.

Run-time Properties

The run-time properties include properties that the Data Integration Service uses when writing data to the target at run time, such as reject file names and directories.

The run-time property for Azure Event Hub targets includes the name of the Azure Event Hub connection.

Advanced Properties

The following table describes the advanced properties for Azure Event Hub targets:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Batch Size	The batch size of the events. Specify an integer value for the batch size.

Property	Description
Shared Access Policy Name	The name of the Event Hub Namespace Shared Access Policy. To write to Event Hubs, you must have Send permissions.
	If you specify a value for this property, it overwrites the value configured in the Azure EventHub connection.
Shared Access Policy Primary Key	The primary key of the Event Hub Namespace Shared Access Policy.
	If you specify a value for this property, it overwrites the value configured in the Azure EventHub connection.
Publisher Properties	The Event Hub publisher configuration properties. Specify properties as key-value pairs. For example, key1=value1,key2=value2

Column Projections Properties

The Developer tool displays the column projection properties in the **Properties** view of the write operation.

To specify column projection properties, double click on the write operation and select the data object. The following table describes the columns projection properties that you configure for Azure EventHub targets:

Property	Description
Column Name	The field in the target that the data object writes to.
	This property is read-only.
Type	The native data type of the target.
	This property is read-only.
Enable Column Projection	Indicates that you use a schema to publish the data to the target.
	By default, the data is streamed in binary format. To change the streaming format, select this option and specify the schema format.
Schema Format	The format in which you stream data to the target. You can select one of the following formats:
	- XML
	- JSON
	- Flat
	- Avro
Use Schema	Specify the XSD schema for the XML format or the sample file for JSON or Avro format. For the Flat file format, configure the schema to associate a flat file.
Column Mapping	The mapping of data object to the target. Click **View** to see the mapping.
Project as Hierarchical Type	Project columns as complex data type for hierarchical data.
	For more information, see the *Data Engineering Integration User Guide*.

Cassandra Data Objects

A Cassandra data object is a physical data object that represents data in an Cassandra resource. After you configure an Cassandra connection, create an Cassandra data object to write to Cassandra targets.

Cassandra is an open source, NoSQL database that is highly scalable and provides high availability. You can use Cassandra to store large amounts of data spread across data centers or when your applications require high write access speed.

In a Cassandra database, a column family is similar to a table in a relational database and consists of columns and rows. Similar to the relational database, each row is uniquely identified by a row key. The column name uniquely identifies each column in the column family. The number of columns in each row can vary, and client applications can determine the number of columns in each row.

You can read, write, and manipulate a group of data by using collections in Cassandra. The Cassandra database supports the following collection types:

- **List**. A list is a collection of one or more ordered elements.
- **Set**. A set is a collection of one or more elements.
- **Map**. A map is a collection of key-value pairs.

You can configure the data object write operation properties that determine how data can be loaded to Cassandra targets. After you create an Cassandra data object, create a write operation. You can use the Cassandra data object write operation as a target in streaming mappings. You can create the data object write operation for the Cassandra data object automatically. Then, edit the advanced properties of the data object write operation and run a mapping.

You can import a Cassandra column family as an ODBC data object and create streaming mappings to write data to Cassandra. You must configure the ODBC driver before you import Cassandra column families. When you run a Cassandra mapping, the Data Integration Service uses the Informatica Cassandra ODBC data source to load data to the Cassandra database. When you run Cassandra mappings, ensure that the optimization level is set to none. If you do not set the optimization level as none, the mappings might fail.

Cassandra Data Object Overview Properties

The Cassandra Overview view displays general information about the Cassandra data object and the object properties that apply to the Cassandra files you import. The Developer tool displays overview properties for Cassandra in the **Overview** view.

General Properties

You can configure the following properties for an Cassandra data object:

- **Name**. Name of the Cassandra data object.
- **Description**. Description of the Cassandra data object.
- **Native Name**. Name of the Cassandra data object.
- **Path Information**. The path of the data object in Cassandra. For example, `/demo/test1/`

  Note: While creating a Cassandra connection, it is not mandatory to provide a value for the **Default Keyspace** attribute. If you define the keyspace value in a Cassandra connection and then create a Cassandra data object, the streaming mapping considers the keyspace value provided in the connection. However, if you provide a different **Path Information** attribute value while creating a Cassandra data object, the path information value overrides all other values and is considered as the default keyspace value.

Column

You can configure the name, native name, data type, precision, access type, scale, and description of the columns in the Cassandra data object.

Cassandra Data Object Write Operation Properties

The Data Integration Service writes data to an Cassandra object based on the data object write operation. The Developer tool displays the data object write operation properties for the Cassandra data object in the **Data Object Operation** section.

You can edit the run-time and advanced properties.

General Properties

The Developer tool displays general properties for Cassandra targets in the **Write** view.

The following table describes the general properties that you view for Cassandra targets:

Property	Description
Name	The name of the Cassandra target write operation. You can edit the name in the **Overview** view. When you use the Cassandra target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Cassandra target write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Cassandra targets:

Property	Description
Name	The name of the port.
Type	The data type of the port.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the port.

Run-time Properties

The run-time properties display the name of the connection used for write transformation.

The following table describes the run-time properties that you configure for an Cassandra write operation:

Property	Description
Connection	Name of the Cassandra connection.
Partition Type	You can configure multiple partitions when you write to a Cassandra target. You can either select **None** or **Dynamic** partition type.
Reject file directory	Directory where the reject file exists. Default is the RejectDir system parameter.
Reject file name	File name of the reject file. Default is <output_file_name>.bad.

Advanced Properties

The following table describes the advanced properties that you configure for Cassandra targets in streaming mappings:

Property	Description
Tracing level	Controls the amount of detail in the mapping log file.
Maintain Row Order	Maintain row order while writing data to the target. Select this option if the Data Integration Service should not perform any optimization that can change the row order. When the Data Integration Service performs optimizations, it might lose the row order that was established earlier in the mapping. You can establish row order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When you configure a target to maintain row order, the Data Integration Service does not perform optimizations for the target.
Truncate Target Table	Truncates the target before loading data.
Update Mode	Defines how rows are updated in the target tables. Select from the following options: - **Update As Update**. Rows matching the selected update columns are updated in the target. - **Update Else Insert**. Rows matching the selected update columns are updated in the target. Rows that do not match are appended to the target.
postSQL	The SQL command the Data Integration Service runs against the target database after it writes to the target. The Developer tool does not validate the SQL.
preSQL	The SQL command the Data Integration Service runs against the target database before it reads the source. The Developer tool does not validate the SQL.

Cassandra, ODBC, and Transformation Data types

When you import a Cassandra column family as a data object, the transformation data types corresponding to the ODBC data types appear in the Developer Tool.

The Informatica Cassandra ODBC driver reads Cassandra data and converts the Cassandra data types to ODBC data types. The Data Integration Service converts the ODBC data types to transformation data types.

The following table lists the Cassandra data types and the corresponding ODBC and transformation data types:

Cassandra Data Types	ODBC Data Types	Transformation Data Types
ASCII	Varchar	Varchar
Bigint	Bigint	Bigint
Blob	Varbinary	Varbinary
Boolean	Bit	Bit
Date	Date	Datetime
Decimal	Decimal	Decimal
Double	Double	Double
Float	Real	Real
Inet	Varchar	Varchar. Cassandra Inet data type supports IP addresses in IPv4 and IPv6 formats. Informatica Cassandra ODBC driver does not support reading or writing IP addresses in IPv6 format.
Int	Integer	Integer
Smallint	Smallint	Integer
Text	Wvarchar	Varchar
Timestamp	Varchar	Time stamp. The Informatica Cassandra ODBC driver treats the Time stamp data type values in Coordinated Universal Time (UTC) format. Range of Timestamp value is 1601-01-01 00:00:00.000 to 9999-12-31 23:59:59.999
Timeuuid	Varchar	Varchar
Tinyint	Tinyint	Integer
Uuid	Varchar	Varchar. When you use comparison operators in data of Uuid data type, enclose values of Uuid data type within single quotation marks. For example, in the following query, the value for the Uuid data type is enclosed within single quotation marks: `DELETE from pc.map_nesting_tgt_vt_col_map2 where pc.map_nesting_tgt_vt_col_map2.col_uuid='152d2030-14e6-11e4-aef1-fd88d8253a43';`

Cassandra Data Types	ODBC Data Types	Transformation Data Types
Varchar	Varchar	Varchar
Varint	Numeric	Numeric

Note: The Varint data type is stored internally as Long data type in the Spark execution plan.

Complex File Data Objects

Create a complex file data object with an HDFS connection to write data to HDFS sequence files or binary files. You can write data to one or more files.

When you create a complex file data object, a read and write operation is created. To use the complex file data object as a target in streaming mappings, configure the complex file data object write operation properties. You can select the mapping environment and run the mappings on the Spark engine of the Hadoop environment.

When you configure the data operation properties, specify the format in which the complex file data object writes data to the HDFS sequence file. You can specify Avro, JSON, XML, or Parquet as file format. You can also specify binary as format.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Complex File Data Object Overview Properties

The Data Integration Service uses overview properties when it reads data from or writes data to a complex file.

Overview properties include general properties that apply to the complex file data object. They also include object properties that apply to the resources in the complex file data object. The Developer tool displays overview properties for complex files in the **Overview** view.

General Properties

The following table describes the general properties that you configure for complex files:

Property	Description
Name	The name of the complex file operation.
Description	The description of the complex file data object.

Property	Description
Native Name	Name of the HDFS connection.
Path Information	The path on the Hadoop file system.

Complex File Header Port

A complex file data object contains a default header port named FileName. Use the FileName port to write events to different target files. At run time, the Data Integration Service creates a folder for each unique FileName port value with one or more target files in the target location.

The number of target files in each folder might vary based on the timestamp values of the events, the partitions to which the events belong, and the executors that process the events.

For example, after you persist your source data in multiple Kafka topics, you want to write the source data to multiple HDFS files. You can use the FileName port of a complex file data object to write the data to multiple HDFS files.

Compression and Decompression for Complex File Targets

You can write compressed files, specify compression formats, and decompress files. You can use compression formats such as Bzip2 and Lz4, or specify a custom compression format.

You can compress sequence files at a record level or at a block level.

For information about how Hadoop processes compressed and uncompressed files, see the Hadoop documentation.

The following table describes the compression formats:

Compression Options	Description
None	The file is not compressed.
Auto	The Data Integration Service detects the compression format of the file based on the file extension.
DEFLATE	The DEFLATE compression format that uses a combination of the LZ77 algorithm and Huffman coding.
Gzip	The GNU zip compression format that uses the DEFLATE algorithm.
Bzip2	The Bzip2 compression format that uses the Burrows–Wheeler algorithm.
Lzo	The Lzo compression format that uses the Lempel-Ziv-Oberhumer algorithm. In a streaming mapping, the compression format is LZ4. The LZ4 compression format uses the LZ77 algorithm.

Compression Options	Description
Snappy	The LZ77-type compression format with a fixed, byte-oriented encoding.
Custom	Custom compression format. If you select this option, you must specify the fully qualified class name implementing the `CompressionCodec` interface in the **Custom Compression Codec** field.

Complex File Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to a complex file. Select the Input transformation to edit the general, ports, targets, and run-time properties.

General Properties

The Developer tool displays general properties for complex file targets in the **Write** view.

The following table describes the general properties that you configure for complex file targets:

Property	Description
Name	The name of the complex file write operation. You can edit the name in the **Overview** view. When you use the complex file as a target in a mapping, you can edit the name in the mapping.
Description	The description of the complex file write operation.

Ports Properties

Port properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for complex file targets:

Property	Description
Name	The name of the resource.
Type	The native data type of the resource.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	The scale for each column. Scale is the maximum number of digits that a column can accommodate to the right of the decimal point. Applies to decimal columns. The scale values you configure depend on the data type.
Detail	The detail of the data type.
Description	The description of the resource.

Target Properties

The target properties list the targets of the complex file data object.

The following table describes the target properties that you configure for complex file targets in a streaming mapping:

Property	Description
Target	The target which the complex data object writes to. You can add or remove targets.

Run-time Properties

The run-time properties include the name of the connection that the Data Integration Service uses to write data to the HDFS sequence file or binary file.

You can configure dynamic partitioning or fixed partitioning.

Advanced Properties

The Developer tool displays the advanced properties for complex file targets in the Input transformation in the **Write** view.

The following table describes the advanced properties that you configure for complex file targets in a streaming mapping:

Property	Description
Operation Type	Indicates the type of data object operation. This is a read-only property.
File Directory	The location of the complex file target. At run time, the Data Integration Service creates temporary directories in the specified file directory to manage the target files. If the directory is in HDFS, enter the path without the node URI. For example, `/user/lib/testdir` specifies the location of a directory in HDFS. The path must be 512 characters or less.
Overwrite Target	Not applicable for streaming mappings.
File Name	The name of the output file. Spark appends the file name with a unique identifier before it writes the file to HDFS.
File Format	The file format. Select one of the following file formats: - Binary. Select Binary to read any file format. - Sequence. Select Sequence File Format for target files of a specific format that contain key and value pairs.
Output Format	The class name for files of the output format. If you select Output Format in the **File Format** field, you must specify the fully qualified class name implementing the `OutputFormat` interface.
Output Key Class	The class name for the output key. By default, the output key class is NullWritable.

Property	Description
Output Value Class	The class name for the output value. By default, the output value class is Text.
Compression Format	Optional. The compression format for binary files. Select one of the following options: - None - Auto - DEFLATE - gzip - bzip2 - LZO - Snappy - Custom
Custom Compression Codec	Required for custom compression. Specify the fully qualified class name implementing the `CompressionCodec` interface.
Sequence File Compression Type	Optional. The compression format for sequence files. Select one of the following options: - None - Record - Block
Stream Rollover Size in GB	Optional. Target file size, in gigabytes (GB), at which to trigger rollover. A value of zero (0) means that the target file does not roll over based on size. Default is 1 GB.
Stream Rollover Time in Hours	Optional. Length of time, in hours, for a target file to roll over. After the time period has elapsed, the target file rolls over. A value of zero (0) means that the target file does not roll over based on time. Default is 1 Hour.
Schema Location	Optional. The schema location to fetch the schema in a streaming mapping. Only Avro schema using binary file format is supported. You must disable the column projection. If you select `External Location` for the dynamic schema strategy, you must create a `writer.avsc` file having the schema content at the schema location and keep it under the topic name. For example: `<Schema Location>/<Topic Name>/writer.avsc`. Then, specify only the path till the schema location. **Note:** Schema location must be named as per the topic name. If **Dynamic Schema Strategy** is enabled at source and schema location is not provided for the HDFS target, then at runtime schema location fetches the schema for the HDFS target from the source transformation schema.
Interim Directory	The active directory location of the complex file target. This directory stages all the files currently in open state. When the stream rollover condition is met, the files are moved from the interim directory to the target directory.

Column Projection Properties

The following table describes the columns projection properties that you configure for complex file targets:

Property	Description
Column Name	The name of the column in the source table that contains data.
Type	The native data type of the resource.

Property	Description
Enable Column Projection	Indicates that you use a schema to publish the data to the target.
	By default, the columns are projected as binary data type. To change the format in which the data is projected, select this option and specify the schema format.
	When you create a complex file for an XML schema, select Binary as the resource format. After you create the object, enable column projection and manually select the schema format as XML.
Schema Format	The format in which you stream data to the target. Select one of the following formats: - XML - JSON - Avro - Parquet
Use Schema	Specify the XSD schema for the XML format, the sample JSON for the JSON format, or sample Avro file for the Avro format.
Column Mapping	Click **View** to see the mapping of the data object to target mapping.
Project as Hierarchical Type	Project columns as complex data type for hierarchical data. For more information on hierarchical data, see the *Data Engineering Integration User Guide*.

Complex File Execution Parameters

When you write to an HDFS complex file, you can configure how the complex file data object writes to the file. Specify these properties in the execution parameters property of the streaming mapping.

Use execution parameters to configure the following properties:

Rollover properties

When you write to an HDFS complex file, the file rollover process closes the current file that is being written to and creates a new file on the basis of file size or time. When you write to the HDFS file, you can configure a time-based rollover or size-based rollover. You can use the following optional execution parameters to configure rollover:

- `rolloverTime`. You can configure a rollover of the HDFS file when a certain period of time has elapsed. Specify rollover time in hours. For example, you can specify a value of 1.

- `rolloverSize`. You can configure a rollover of the HDFS target file when the target file reaches a certain size. Specify the size in GB. The default rollover size is 1 GB.

The default is size-based rollover. You can implement both rollover schemes for a target file, in which case, the event that occurs first triggers a rollover. For example, if you set rollover time to 1 hour and rollover size to 1 GB, the target service rolls the file over when the file reaches a size of 1 GB even if the 1-hour period has not elapsed.

Pool properties

You can configure the maximum pool size that one Spark executor can have to write to a file. Use the `pool.maxTotal` execution parameter to specify the pool size. Default pool size is 8.

Retry Interval

You can specify the time interval for which Spark tries to create the target file or write to it if it fails to do so the first time. Spark tries a maximum of three times during the time interval that you specify. Use the

`retryTimeout` execution parameter to specify the timeout in milliseconds. Default is 30,000 milliseconds.

Temporary Directory for the Complex File Target

At run time, the Data Integration Service creates a temporary directory in the location that you set for the **File Directory** property that you configure for the complex file targets in a streaming mapping. A temporary directory is created for each mapping. A temporary directory contains the target files to which the data is currently written.

Based on the rollover limit, data is written to the target file in the temporary directory. After a target file reaches the specified rollover limit, the target file is closed and moved to the specified target file directory from the temporary directory.

For example, if you set the **File Directory** property to `hdfs://demo.domain.com:8020/tmp/batch/`, you can find the target file to which data is currently written in the following directory: `hdfs://demo.domain.com:8020/tmp/batch/active/586bdfd67170008d125f52f68c1f02/`. The `hdfs://demo.domain.com:8020/tmp/batch/` directory contains the target files that are moved after the rollover limit is reached.

If a mapping fails, all the target files are closed and moved to the specified target file directory from the temporary directory, and the temporary directory will be deleted. In the preceding example, the temporary directory `586bdfd67170008d125f52f68c1f02` will be deleted.

If you use Data Engineering Integration to process batch data, you can use the batch complex files that are created in the file directory as the source objects in Data Engineering Integration.

FileName Port in Complex File

When you create a data object read or write operation for HDFS complex files, the FileName port appears by default.

When the Spark engine writes to HDFS complex files using a FileName port, it uses the following process to write the data:

1. At run time, the Data Integration Service creates separate directories for each value in the FileName port and adds the target files within the directories.

2. The file rollover process closes the current file to which data is being written to and creates a new file based on the configured rollover value.
   you can configure the following optional execution rollover parameters at the design-time based on time and size:

   - Stream Rollover Time in Hours. Specify the rollover time in hours for a target file when a certain period of time has elapsed.
   - Stream Rollover Size in GB. Specify the size in GB for a target file when the target file reaches a certain size.

3. When a target file reaches the configured rollover value, the target file is rolled over and moved to the specified HDFS Complex File target location.

4. The Spark engine creates sub-directories in the specified HDFS comples file target location for each value in the FileName port.

5. The Spark engine moves the rolled over target files to the sub-directories created for each value in the FileName port in the specified Complex file target location.

You can configure both rollover schemes for an HDFS complex file target. The Spark engine rolls over to based on the first event that triggers. For example, if you configure rollover time to 1 hour and rollover size to

1 GB, the target service rolls the file over when the file reaches a size of 1 GB even if the 1 hour period has not elapsed.

Confluent Kafka Data Objects

A Confluent Kafka data object is a physical data object that represents data in a Kafka stream or a Confluent Kafka stream. After you configure a Messaging connection, create a Confluent Kafka data object to write data to Kafka brokers or Confluent Kafka brokers using schema registry.

Confluent Kafka runs as a cluster comprised of one or more servers each of which is called a broker. Confluent Kafka brokers stream data in the form of messages. These messages are published to a topic.

Confluent Kafka topics are divided into partitions. The Spark engine can write to the partitions of the topics in parallel to achieve better throughput and to scale the number of messages processed. Message ordering is guaranteed only within partitions. For optimal performance you should have multiple partitions.

Write Operation in Confluent Kafka

You can use the Confluent Kafka data object write operation as a target in streaming mappings. By default, the write operation is created for Confluent Kafka.

File Format in Confluent Kafka

When you configure the data operation properties, specify the format in which the Confluent Kafka data object writes data.

You can specify XML, JSON, Avro, or Flat as format for Kafka data objects. When you specify XML format, you must provide a XSD file. When you specify JSON or Flat format, you must provide a sample file. When you specify Avro format, provide a sample Avro schema in an .avsc file.

You can specify Avro as the format for Confluent Kafka data objects using schema registry.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Note: In Databricks environment, a streaming mapping fails when you enable schema registry in the connection properties of the Confluent Kafka data object.

Confluent Kafka Data Object Overview Properties

You can use the Developer tool to view the overview properties for Confluent Kafka messages in the Overview view.

The overview properties include general properties, object properties, and column properties that apply to the Confluent Kafka data object.

General Properties

The following table describes the general properties that you configure for Confluent Kafka data object:

Property	Description
Name	Name of the Confluent Kafka data object.
Description	Description of the Confluent Kafka data object.
Connection	Name of the Confluent Kafka connection.

Objects Properties

The following table describes the objects properties that you configure for Confluent Kafka data object:

Property	Description
Name	Name of the topic or topic pattern of the Confluent Kafka data object.
Description	Description of the Confluent Kafka data object.
Native Name	Native name of Confluent Kafka data object.
Path Information	Type and name of the topic or topic pattern of the Confluent Kafka data object.

Column Properties

The following table describes the column properties that you configure for Confluent Kafka data object:

Property	Description
Name	Name of the Confluent Kafka data object.
Native Name	Native name of the Confluent Kafka data object.
Type	Native data type of the Confluent Kafka data object.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	Scale of the data type.
Description	Description of the Confluent Kafka data object.

Confluent Kafka Header Ports

The Confluent Kafka data object contains default header ports that represent metadata associated with events.

The following table describes the header ports of the Confluent Kafka data object:

Header ports	Data type	Description
partitionId	integer	Partition ID of the data. When the Confluent Kafka topic is divided into partitions, a sequential ID is assigned to each event and is unique to each event within the partition. Use the partition ID to identify the event that you want to consume from a particular partition.
key	binary	Key value associated with an event. You can group events based on the key value and then process the data.
TopicName	string	Name of the Confluent Kafka topic from where you receive events.

Confluent Kafka Data Object Write Properties

The Data Integration Service uses write operation properties when it writes data to a Confluent Kafka broker.

General Properties

The general properties display the name and description of the Confluent Kafka sources.

The following table describes the general properties that you view for Confluent Kafka targets:

Property	Description
Name	Name of the Confluent Kafka broker write operation. You can edit the name in the **Overview** view. When you use the Confluent Kafka broker as a target in a mapping, you can edit the name in the mapping.
Description	Description of the Confluent Kafka broker write operation.

Ports Properties

The ports properties display the port names and port attributes such as data type and precision of the Confluent Kafka targets.

The following table describes the port properties that you configure for Confluent Kafka broker targets:

Property	Description
Name	Name of the target header field.
Type	Native data type of the resource.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	Detail of the data type.
Scale	Scale of the data type.
Description	Description of the resource.

Targets Properties

The targets properties display the targets of the Confluent Kafka data object.

The following table describes the targets property that you can configure for Confluent Kafka targets:

Property	Description
Target	The targets that the Confluent Kafka data object writes to. You can add or remove targets.

Schema Properties

The schema properties display the column related details of the Confluent Kafka targets. To specify schema properties, select the data operation and enable the column projection.

The following table describes the schema properties that you configure for Confluent Kafka targets:

Property	Description
Column Name	Name field that contains data. This property is read-only.
Type	Native data type of the source. This property is read-only.
Enable Column Projection	Indicates that you use a schema format to write the data.
Schema Format	The format in which the data is written. Confluent Kafka using schema registry supports Avro format. For Kafka, select one of the following formats: - XML - JSON - Flat - Avro
Schema	Specify the XSD schema for the XML format, a sample file for JSON, or .avsc file for Avro format. For the Flat file format, configure the schema to associate a flat file to the Kafka target. When you provide a sample file, the Data Integration Service uses UTF-8 code page when writing the data.
Column Mapping	Mapping of target data to the data object. Click **View** to see the mapping.

Run-time Properties

The run-time properties display the name of the connection.

The following table describes the run-time property that you configure for Confluent Kafka targets:

Property	Description
Connection	Name of the Confluent Kafka connection.

Advanced Properties

The advanced properties display the details about operation type and time after which the Confluent Kafka target starts writing Confluent Kafka messages to a Confluent Kafka topic.

The following table describes the advanced properties that you configure for Confluent Kafka targets:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Metadata Fetch Timeout in milliseconds	Time after which the metadata is not fetched.
Batch Flush Time in milliseconds	Interval after which the data is published to the target.
Batch Flush Size in bytes	Batch size of the events after which the data is written to the target.

Producer Configuration Properties	Configuration properties for the producer. If the Confluent Kafka data object is writing data to a Confluent Kafka cluster that is configured for Kerberos authentication, include the following property:
	`security.protocol=SASL_PLAINTEXT,sasl.kerberos.service.name=kafka,`
	`sasl.mechanism=GSSAPI`

Google Cloud Storage Data Objects

A Google Cloud Storage data object is a physical data object that represents data in a Google Cloud Storage resource. After you configure a Google Cloud Storage connection, create a Google Cloud Storage data object to write to Google Cloud Storage targets.

You can configure the data object write operation properties that determine how data can be loaded to Google Cloud Storage targets. After you create a Google Cloud Storage data object, create a write operation. You can use the Google Cloud Storage data object write operation as a target in streaming mappings. You can create the data object write operation for the Google Cloud Storage data object automatically. Then, edit the advanced properties of the data object write operation and run a mapping.

When you configure the data operation properties, specify the format in which the data object writes data. You can specify Avro, JSON, or Parquet as format. When you specify Avro format, provide a sample Avro schema in an `.avsc` file. When you specify JSON, you must provide a sample file.

You can pass any payload format directly from source to target in streaming mappings. To successfully run a streaming mapping when you select multiple objects from different Google Cloud Storage buckets, ensure that all the Google Cloud Storage buckets belong to the same region and use the same credentials to access the Google Cloud Storage buckets. However, the cross-account service is not enabled for a Google Cloud Storage target.

When you use a Google Cloud Storage target in a streaming mapping, you can configure size-based rollover and time-based rollover while creating the mapping.

Note: You cannot use the Mapping Flow option in a streaming mapping with Google Cloud Storage target.

Google Cloud Storage Data Object Overview Properties

The Google Cloud Storage Overview view displays general information about the Google Cloud Storage data object and the object properties that apply to the Google Cloud Storage files you import. The Developer tool displays overview properties for Google Cloud Storage in the **Overview** view.

General Properties

The following table describes the general properties that you configure for Google Cloud Storage data objects:

Property	Description
Name	Name of the Google Cloud Storage data object.
Description	Description of the Google Cloud Storage data object.
Native Name	Name of the Google Cloud Storage data object.
Path Information	The path of the data object in Google Cloud Storage. For example, `/gcs.demo/test1/`

Column Properties

The following table describes the column properties that you can view for Google Cloud Storage data objects:

Property	Description
Name	Name of the column.
Native Name	Native name of the data type of the column.
Type	Data type of the column.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Access Type	The access type of the port or column.
Description	Description of the column.

Google Cloud Storage Data Object Write Operation Properties

The Data Integration Service writes data to a Google Cloud Storage object based on the data object write operation. The Developer tool displays the data object write operation properties for the Google Cloud Storage data object in the **Data Object Operation** section.

You can edit the format, run-time, and advanced properties.

General Properties

The Developer tool displays general properties for Google Cloud Storage targets in the **Write** view.

The following table describes the general properties that you view for Google Cloud Storage targets:

Property	Description
Name	The name of the Google Cloud Storage target write operation. You can edit the name in the **Overview** view. When you use the Google Cloud Storage target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Google Cloud Storage target write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Google Cloud Storage targets:

Property	Description
Name	The name of the port.
Type	The data type of the port.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the port.

Schema Properties

The schema properties display the column related details of the Google Cloud Storage targets. To specify schema properties, select the data operation and enable the column projection.

The following table describes the schema properties that you configure for Google Cloud Storage targets:

Property	Description
Column Name	Name field that contains data. This property is read-only.
Column Type	Native data type of the source. This property is read-only.
Enable Column Projection	Indicates that you use a schema format to write the data.

Property	Description
Schema Format	The format in which the data is written. Select one of the following formats: - Avro - JSON - Parquet When you run mappings on the Spark engine, you cannot write data to Google Cloud Storage in binary or flat file format.
Schema	Specify .avsc file for Avro format, sample file for JSON, or Parquet file format.
Column Mapping	Mapping of target data to the data object. Click **View** to see the mapping.

Run-time Properties

The run-time properties display the name of the connection used for write transformation.

The following table describes the run-time properties that you configure for a Google Cloud Storage write operation:

Property	Description
Connection	Name of the Google Cloud Storage connection.
Partition Type	Not applicable for streaming mappings.
Reject file directory	Directory where the reject file exists.
Reject file name	File name of the reject file.

Advanced Properties

Google Cloud Storage data object write operation properties include run-time properties that apply to the Google Cloud Storage data object.

The Developer tool displays advanced properties for the Google Cloud Storage data object operation in the Advanced view.

The following table describes the Advanced properties for a Google Cloud Storage data object write operation:

Property	Description
Tracing Level	Amount of detail that appears in the log. You can choose from terse, normal, verbose initialization, or verbose data. Default is normal.
Maintain row order	Maintain row order while writing data to the target. Select this option if the Data Integration Service should not perform any optimization that can change the row order. When the Data Integration Service performs optimizations, it might lose the row order that was established earlier in the mapping. You can establish row order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When you configure a target to maintain row order, the Data Integration Service does not perform optimizations for the target.
Google Cloud Storage Path	Overrides the Google Cloud Storage path that you specified in the connection. This property is required when the source is not a flat file. Use the following format: `gs://<bucket name>` or `gs://<bucket name>/<folder name>`
Target File Name	Name of the Google Cloud Storage target file that you want to write to. The file specified at run time overrides the file specified while creating a connection.
Target Schema Strategy	Not applicable for streaming mappings.

HBase Data Objects

An HBase data object is a physical data object that represents data in an HBase resource. After you create an HBase connection, create an HBase data object with a write data operation to write data to an HBase table.

When you create an HBase data object, you can select an HBase table and view all the column families in the table. You can specify the column names in the column family if you know the column name and data type, or you can search the rows in the HBase table and specify the columns.

You can write to a column family or to a single binary column. When you create the data object, specify the column families to which you can write or choose to write all the data as a single stream of binary data.

Data Object Column Configuration

When you want to write data to columns in a column family, you can specify the columns when you create the HBase data object.

You can write data to columns in one of the following ways:

- Add the columns in the column families.
- Search for the columns names in the column family and add the columns.
- Get all the columns in a column family as a single stream of binary data.

When you manually create the data object, the column name should be of the following format:
`columnFamily__columnQualifier`

Add Columns

When you create a data object, you can specify the columns in one or more column families in an HBase table.

When you add an HBase table as the resource for an HBase data object, all the column families in the HBase table appear. If you know the details of the columns in the column families, you can select a column family and add the column details. In the **Column Families** dialog box, select the column family to which you want to add the columns. Column details include column name, data type, precision, and scale.

Although data is stored in binary format in HBase tables, you can specify the associated data type of the column to transform the data. To avoid data errors or incorrect data, verify that you specify the correct data type for the columns.

Verify that you specify valid column details when you add columns to avoid unexpected run-time behaviors. If you do not specify a value for a column when you write data to an HBase table, the Data Integration Service specifies a null value for the column at run time.

If the HBase table has more that one column family, you can add column details for multiple column families when you create the data object. Select one column family at a time and add the columns details. The column family name is the prefix for all the columns in the column family for unique identification.

Search and Add Columns

When you create a data object, you can search the rows in an HBase table to identify the column in the table and select the columns you want to add.

When you do not know the columns in an HBase table, you can search the rows in the table to identify all the columns and the occurrence percentage of the column. You can infer if the column name is valid based on the number of times the column occurs in the table. For example, if column name eName occurs rarely while column name empName occurs in a majority of rows, you can infer the column name as empName.

When you search and add columns, you can specify the maximum number of rows to search and the occurrence percentage value for a column. If you specify the maximum numbers of rows as 100 and the column occurrence percent as 90, all columns that appear at least 90 times in 100 rows appear in the results. You can select the columns in the results to add the columns to the data object.

Get All Columns

Binary data or data that can be converted to a byte array can be stored in an HBase column. You can read from and write to an HBase tables in bytes.

When you create a data object, you can choose to get all the columns in a column family as a single stream of binary data.

Use the HBase data object as a target to write data in all the columns in the source data object as a single column of binary data in the target HBase table.

The Data Integration Service generates the data in the binary column based on the protobuf format. Protobuf format is an open source format to describe the data structure of binary data. The protobuf schema is described as messages.

HBase Object Overview Properties

The Data Integration Service uses overview properties when it writes data to an HBase resource.

Overview properties include general properties that apply to the HBase data object. They also include object properties that apply to the resources in the HBase data object. The Developer tool displays overview properties for HBase resources in the Overview view.

General Properties

The following table describes the general properties that you configure for the HBase data objects:

Property	Description
Name	Name of the HBase data object.
Location	The project or folder in the Model repository where you want to store the HBase data object.
Native Name	Native name of the HBase data object.
Path	Path to the HBase data object.

Add Column Properties

In the **Column Families** dialog box, select the column family to which you want to add the columns. The following table describes the column properties that you configure when you associate columns with column families:

Property	Description
Name	Name of the column in the column family.
Type	Data type of the column.
Precision	Precision of the data.
Scale	Scale of the data.

HBase Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to an HBase resource.

HBase data object write operation properties include run-time properties that apply to the HBase data object.

Advanced Properties

The Developer tool displays the advanced properties for HBase targets in the **Advanced** view.

The following table describes the advanced properties that you can configure for an HBase data object in a streaming mapping:

Property	Description
Operation Type	The type of data object operation. This is a read-only property.
Date Time Format	Format of the columns of the date data type. Specify the date and time formats by using any of the Java date and time pattern strings.
Auto Flush	Optional. Indicates whether you want to enable Auto Flush. You can set auto flush to the following values: - Enable **Auto Flush** to set the value to true. The Data Integration Service runs each Put operation immediately as it receives them. The service does not buffer or delay the Put operations. Operations are not retried on failure. When you enable auto flush, the operations are slow as you cannot run operations in bulk. However, you do not lose data as the Data Integration Service writes the data immediately. - Disable **Auto Flush** to set the auto flush value to false. When you disable auto flush, the Data Integration Service accepts multiple Put operations before making a remote procedure call to perform the write operations. If the Data integration Service stops working before it flushes any pending data writes to HBase, that data is lost. Disable auto flush if you need to optimize performance. Default is disabled.
default Column Data Type	Not applicable for streaming mappings.
default Precision	Not applicable for streaming mappings.
default Scale	Not applicable for streaming mappings.
Default Column Family	Not applicable for streaming mappings.
Control File Location	Not applicable for streaming mappings.

JMS Data Objects

A JMS data object is a physical data object that accesses a JMS server. After you configure a JMS connection, create a JMS data object to write to JMS targets.

JMS providers are message-oriented middleware systems that send JMS messages. The JMS data object connects to a JMS provider to write data.

The JMS data object can write JMS messages to a JMS provider. When you configure a JMS data object, configure properties to reflect the message structure of the JMS messages. The input ports and output ports are JMS message headers.

When you configure the write data operation properties, specify the format in which the JMS data object writes data. You can specify XML, JSON, or Flat as format. When you specify XML format, you must provide an XSD file. When you specify JSON or Flat format, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Integration with JMS

You manually create JMS source and target data objects to reflect the message structure of JMS messages.

The JMS data object can read messages of type TextMessage. This type of message contains a string object. TextMessages can contain XML or JSON message data.

JMS Message Structure

JMS messages contain the following components:

- Header
- Properties
- Body

Header Fields

JMS messages contain a fixed number of header fields. Each JMS message uses these fields regardless of message type. Every JMS source and target definition includes a pre-defined set of header fields.

The following table describes the JMS message header fields:

Header field	Data type	Description
JMSDestination	String	Destination to which the message is sent. JMS destinations can be a message queue or a recipient who listens for messages based on the message topic.
JMSDeliveryMode	Integer	Delivery mode of the message. The delivery mode can be persistent or non-persistent.
JMSMessageID	String	Unique identification value for the message.
JMSTimestamp	Date/Time	Time at which the message was handed off to the provider to be sent to the destination.
JMSCorrelationID	String	Links one message with another. For example, JMSCorrelationID can link a response message with the corresponding request message.
JMSReplyTo	String	Destination to which a reply message can be sent.
JMSRedelivered	String	Indicates that a message might have been delivered previously, but not acknowledged.

Header field	Data type	Description
JMSType	String	Type of message based on a description of the message. For example, if a message contains a stock trade, the message type might be stock trade.
JMSExpiration	Bigint	Amount of time in milliseconds the message remains valid. The messages remain in memory during this period.
JMSPriority	Integer	Priority of the message from 0-9. 0 is the lowest priority. 9 is the highest.

Property Fields

JMS source and target definitions can optionally include message property fields. Property fields contain additional message header information. JMS providers use properties in a JMS message to give provider-specific information. Applications that use a JMS provider can add property fields with application-specific information to a message.

Body Fields

JMS source and target definitions can optionally include a message body. The body contains one or more fields. Only certain types of JMS messages contain a body.

JMS Data Object Overview Properties

The Data Integration Service uses overview properties when it reads data from a JMS source.

Overview properties include general properties that apply to the JMS data object. They also include object properties that apply to the resources in the JMS data object. The Developer tool displays overview properties for JMS messages in the Overview view.

General Properties

The following table describes the general properties that you configure for JMS data objects:

Property	Description
Name	The name of the JMS data object.
Description	The description of the JMS data object.
Connection	The name of the JMS connection.

Objects Properties

The following table describes the objects properties that you configure for JMS data objects:

Property	Description
Name	The name of the topic or queue of the JMS source.
Description	The description of the JMS source.

Property	Description
Native Name	The native name of JMS source.
Path Information	The type and name of the topic or topic pattern of the JMS source.

JMS Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to a JMS source. You can edit the format, run-time, and advanced properties.

General Properties

The Developer tool displays general properties for JMS targets in the **Write** view.

The following table describes the general properties that you view for JMS targets:

Property	Description
Name	The name of the JMS target write operation. You can edit the name in the **Overview** view. When you use the JMS target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the JMS target write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for JMS targets:

Property	Description
Name	The name of the JMS target.
Type	The native data type of the target.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Target Properties

The target properties list the resources of the JMS data object. You can add or remove resources in the data object.

Run-time Properties

The run-time properties displays the name of the connection.

The following table describes the run-time property that you configure for JMS targets:

Property	Description
Connection	Name of the JMS connection.

Advanced Properties

The Developer tool displays the advanced properties for JMS targets in the Input transformation in the **Write** view.

You can configure the following advanced properties for JMS targets:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Destination	Name of the queue or topic to which the JMS provider publishes messages. The data object subscribes to JMS messages from this queue or topic.
Message Type	The format in which the message is written to the target. Specify one of the following formats: - Text - Binary

Column Projections Properties

The following table describes the columns projection properties that you configure for JMS targets:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.
Enable Column Projection	Indicates that you use a schema to publish the data to the target. By default, the data is streamed in binary format. To change the format in which the data is streamed, select this option and specify the schema format.

Property	Description
Schema Format	The format in which you stream data to the target. You can select one of the following formats: - XML - JSON - Flat
Use Schema	Specify the XSD schema for the XML format, and the sample JSON for the JSON format. For the Flat file format, configure the schema to associate a flat file.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.

Kafka Data Objects

A Kafka data object is a physical data object that represents data in a Kafka stream. After you configure a Messaging connection, create a Kafka data object to write to Apache Kafka brokers.

Kafka runs as a cluster comprised of one or more servers each of which is called a broker. Kafka brokers stream data in the form of messages. These messages are published to a topic. When you write data to a Kafka messaging stream, specify the name of the topic that you publish to. You can also write to a Kerberised Kafka cluster.

Kafka topics are divided into partitions. Spark Structured Streaming can read the partitions of the topics in parallel. This gives better throughput and could be used to scale the number of messages processed. Message ordering is guaranteed only within partitions. For optimal performance you should have multiple partitions.

After you create a Kafka data object, create a write operation. You can use the Kafka data object write operation as a target in Streaming mappings. If you want to configure high availability for the mapping, ensure that the Kafka cluster is highly available.

When you configure the data operation properties, specify the format in which the Kafka data object writes data. You can specify XML, JSON, Avro, or Flat as format. When you specify XML format, you must provide a XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON or Flat format, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

For more information about Kafka clusters, Kafka brokers, and partitions see http://kafka.apache.org/11/documentation.html.

Note: If you use a Kafka data object in a streaming mapping, you cannot use a MapR Streams data object in the same mapping.

Kafka Combinations Support Matrix

The following table shows the different authentication mechanisms that you can use against listener protocols in a streaming mapping with a Kafka target:

Protocols	Authentication Mechanisms		
	Plain	**GSSAPI**	**None**
SSL+TLS1.0	Yes	Yes	Yes
SSL+TLS1.1	Yes	Yes	Yes
SSL+TLS1.2	Yes	Yes	Yes
SSL+TLS1.0,TLS1.1,TLS1.2	Yes	Yes	Yes
SASL_SSL+TLS1.0	Yes	Yes	NA
SASL_SSL+TLS1.1	Yes	Yes	NA
SASL_SSL+TLS1.2	Yes	Yes	NA
SASL_SSL+TLS1.0,TLS1.1,TLS1.2	Yes	Yes	NA

Kafka Data Object Overview Properties

The Data Integration Service uses overview properties when it reads data from or writes data to a Kafka broker.

Overview properties include general properties that apply to the Kafka data object. They also include object properties that apply to the resources in the Kafka data object. The Developer tool displays overview properties for Kafka messages in the **Overview** view.

General Properties

The following table describes the general properties that you configure for Kafka data objects:

Property	Description
Name	The name of the Kafka read operation.
Description	The description of the Kafka data object.
Connection	The name of the Kafka connection.

Objects Properties

The following table describes the objects properties that you configure for Kafka data objects:

Property	Description
Name	The name of the topic or topic pattern of the Kafka data object.
Description	The description of the Kafka data object.
Native Name	The native name of Kafka data object.
Path Information	The type and name of the topic or topic pattern of the Kafka data object.

Column Properties

The following table describes the column properties that you configure for Kafka data objects:

Property	Description
Name	The name of the Kafka data object.
Native Name	The native name of the Kafka data object.
Type	The native data type of the Kafka data object.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	The scale of the data type.
Description	The description of the Kafka data object.
Access Type	The type of access the port or column has.

Kafka Header Ports

The Kafka data object contains default header ports that represent metadata associated with events.

The following table lists the header ports of the Kafka data object:

Header ports	Data type	Description
partitionId	integer	Partition ID of the data. When the Kafka topic is divided into partitions, a sequential ID is assigned to each event and is unique to each event within the partition. Use the partition ID to identify the event that you want to consume from a particular partition.
key	binary	Key value associated with an event. You can group events based on the key value and then process the data.
TopicName	string	Name of the Kafka topic from where you receive events.

Kafka Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to a Kafka broker.

General Properties

The Developer tool displays general properties for Kafka targets in the **Write** view.

The following table describes the general properties that you view for Kafka targets:

Property	Description
Name	The name of the Kafka broker write operation. You can edit the name in the **Overview** view. When you use the Amazon S3 target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Kafka broker write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Kafka broker sources:

Property	Description
Name	The name of the source header field.
Type	The native data type of the resource.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the resource.

Run-time Properties

The run-time properties displays the name of the connection.

The following table describes the run-time property that you configure for Kafka targets:

Property	Description
Connection	Name of the Kafka connection.

Target Properties

The targets properties list the targets of the Kafka data object.

The following table describes the sources property that you can configure for Kafka targets:

Property	Description
Target	The target which the Kafka data object writes to. You can add or remove targets.

Advanced Properties

The Developer tool displays the advanced properties for Kafka targets in the Input transformation in the **Write** view.

The following table describes the advanced properties that you configure for Kafka targets:

Property	Description
Operation Type	Specifies the type of data object operation. This is a read-only property.
Metadata Fetch Timeout in milliseconds	The time after which the metadata is not fetched.
Batch Flush Time in milliseconds	The interval after which the data is published to the target.
Batch Flush Size in bytes	The batch size of the events after which the data is written to the target.

Producer Configuration Properties	The configuration properties for the producer. If the Kafka data object is writing data to a Kafka cluster that is configured for Kerberos authentication, include the following property:
	`security.protocol=SASL_PLAINTEXT,sasl.kerberos.service.name=kafka,`
	`sasl.mechanism=GSSAPI`

For more information about Kafka broker properties, see http://kafka.apache.org/082/documentation.html.

Column Projections Properties

The Developer tool displays the column projection properties in the **Properties** view of the write operation.

To specify column projection properties, double click on the write operation and select the data object. The following table describes the columns projection properties that you configure for Kafka targets:

Property	Description
Column Name	The field in the target that the data object writes to. This property is read-only.
Type	The native data type of the target. This property is read-only.

Property	Description
Enable Column Projection	Indicates that you use a schema to publish the data to the target.
	By default, the data is streamed in binary format. To change the streaming format, select this option and specify the schema format.
Schema Format	The format in which you stream data to the target. You can select one of the following formats: - XML - JSON - Flat - Avro
Use Schema	Specify the XSD schema for the XML format, acsample file for JSON, or .avsc file for Avro format.
	For the Flat file format, configure the schema to associate a flat file to the Kafka target. When you provide a sample file, the Data Integration Service uses UTF-8 code page when writing the data.
Column Mapping	The mapping of data object to the target. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for hierarchical data.
	For more information, see the *Data Engineering Integration User Guide*.

Kudu Data Objects

A Kudu data object is a physical data object that represents Kudu as a target. After you configure a Messaging connection, create a data object write operation based on the Kudu data object.

Kudu is a columnar storage manager developed for the Apache Hadoop platform. You can use Kudu to store data in tables. Kudu has a simple data model, where the Kudu table has a primary key made up of one or more columns, each with a defined type. You can use Kudu for fast data search, updates and inserts and to perform real-time analytics on fast data.

A Kudu data object is the representation of data that is based on a Kudu object. Configure the write operation properties to determine how the Data Integration Service must write data to the Kudu. Add the write operation as a Write transformation in a mapping.

You can use Kudu objects as targets in mappings. You can validate and run mappings on the Spark engine in the Hadoop environment.

Kudu Data Object Overview Properties

The Data Integration Service uses overview properties when it writes data to a Kudu data object.

The Overview properties include general properties that apply to the Kudu data object. They also include column properties that apply to the resources in the Kudu data object.

General Properties

The following table describes the general properties that you configure for Kudu data objects:

Property	Description
Name	Name of the Kudu data object.
Description	Description of the Kudu data object.

Column Properties

The following table describes the column properties that you can view for Kudu data objects:

Property	Description
Name	Name of the column.
Native Type	Native data type of the column.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Description	Description of the column.
Nullable	Indicates if the column can contain null values.
Primary Key	Identifies a primary key column in a table that you import from a database source.
Access Type	The access type of the port or column.

Kudu Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to Kudu.

General Properties

The Developer tool displays general properties for Kudu targets in the **Write** view.

The following table describes the general properties that you view for Kudu targets:

Property	Description
Name	The name of the Kudu target write operation. You can edit the name in the **Overview** view. When you use the Kudu target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Kudu target write operation.

Ports Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Kudu targets:

Property	Description
Name	The name of the port.
Type	The data type of the port.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the port.

Run-time Properties

The run-time properties display the name of the connection used for write transformation.

The following table describes the run-time properties that you configure for an Kudu write operation:

Property	Description
Connection	Name of the Kudu connection.

Advanced Properties

Kudu data object write operation properties include run-time properties that apply to the Kudu data object.

The Developer tool displays advanced properties for the Kudu data object operation in the Advanced view.

The following table describes the Advanced properties for a Kudu data object write operation:

Property	Description
Tracing Level	Amount of detail that appears in the log. You can choose from terse, normal, verbose initialization, or verbose data. Default is normal.
Maintain row order	Maintain row order while writing data to the target. Select this option if the Data Integration Service should not perform any optimization that can change the row order.
	When the Data Integration Service performs optimizations, it might lose the row order that was established earlier in the mapping. You can establish row order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When you configure a target to maintain row order, the Data Integration Service does not perform optimizations for the target.

Property	Description
Number of Replica	Sets the number of replicas that each table will have. Replication factor must be odd. Default is 1.
Table Name	Name of the target table.
Table Partitioning Strategy	Type of partition applicable to the table. Partitioning is applicable to primary key fields only. You can select from the following options: - HASH - RANGE Default is HASH. **Note:** You cannot specify the range conditions for the columns of your choice.
Upsert	Upserts any record flagged as update. Select this property when you use the Update Strategy transformation in the mapping to flag records as update. Not applicable for streaming mappings.
Target Schema Strategy	Target schema strategy for the Kudu target table. You can select one of the following target schema strategies: - RETAIN. Retain existing target schema. - CREATE. Create or replace table at run time. When you select the CREATE option, you must provide the value of the Schema Name property to run the mapping successfully. - Assign Parameter. You can assign a parameter to represent the value for the target schema strategy and then change the parameter at run time.

MapR Streams Data Objects

A MapR Streams data object is a physical data object that represents data in a MapR Stream. After you create a MapR Streams connection, create a MapR Streams data object to write data to MapR Streams.

Before you create and use MapR Stream data objects in Streaming mappings, complete the required prerequisites.

For more information about the prerequisite tasks, see the *Data Engineering Integration Guide*.

When you write data to a MapR stream, specify the name of the stream that you publish to.

After you create a MapR Streams data object, create a write data object operation. You can then add the data object write operation as a target in Streaming mappings.

When you configure the data operation properties, specify the format in which the MapR Streams data object writes data. You can specify XML, JSON, or Avro as format. When you specify XML format, you must provide an XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON, you must provide a sample file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Note: If you use a MapR Streams data object in a streaming mapping, you cannot use an Apache Kafka data object in the same mapping.

MapR Streams Object Overview Properties

Overview properties include general properties that apply to the MapR Streams data object. The Developer tool displays overview properties in the Overview view.

General

Property	Description
Name	Name of the MapR Streams data object.
Description	Description of the MapR Streams data object.
Native Name	Native name of the MapR Stream.
Path Information	The path of the MapR Stream.

Column Properties

The following table describes the column properties that you configure for MapR Streams data objects:

Property	Description
Name	The name of the MapR Streams data object.
Native Name	The native name of the MapR Streams data object.
Type	The native data type of the MapR Streams data object.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	The scale of the data type.
Description	The description of the MapR Streams data object.
Access Type	The access type of the port or column.

MapR Streams Header Ports for a Target

The MapR Streams data object contains default header ports that represent metadata associated with events.

The following table lists the header ports of the MapR Streams data object when you use it as a target:

Header ports	Data type	Description
partitionId	integer	Partition ID of the data. When a stream is divided into partitions, a sequential ID is assigned to each event and is unique to each event within the partition. Use the partition ID to identify the event that you want to consume from a particular partition.
key	binary	Key value associated with an event. You can group events based on the key value and then process the data.
TopicName	string	Name of the stream to which you publish events in the following format: `/<any existing stream>:<topicName>` In case the `<topicName>` is not available in the existing stream, the topic will be automatically created for the data to be published to it.

MapR Streams Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to MapR Streams.

General Properties

The Developer tool displays general properties for MapR Stream targets in the **Write** view.

The following table describes the general properties that you view for MapR Stream targets:

Property	Description
Name	The name of the MapR Stream target write operation. You can edit the name in the **Overview** view. When you use the Amazon S3 target as a target in a mapping, you can edit the name in the mapping.
Description	The description of the MapR Stream target write operation.

Port Properties

Ports properties for a physical data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for MapR Stream targets:

Property	Description
Name	The name of the target.
Type	The native data type of the target.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Detail	The detail of the data type.

Property	Description
Scale	The scale of the data type.
Description	The description of the target.

Target Properties

The targets properties list the targets of the MapR Streams data object.

The following table describes the sources property that you can configure for MapR Stream targets:

Property	Description
Target	The target which the MapR Streams data object writes to. You can add or remove targets.

Run-time Properties

The run-time properties displays the name of the connection.

The run-time property for MapR Stream target includes the name of the MapR Stream connection.

Column Projection Properties

The following table describes the columns projection properties that you configure for MapR Stream targets:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the resource. This property is read-only.
Enable Column Projection	Indicates that you use a schema to write data to the target. By default, the data is streamed in binary format. To change the format in which the data is streamed, select this option and specify the schema format.
Schema Format	The format in which the source streams data. You can select one of the following formats: - XML - JSON - Avro
Use Schema	Specify the XSD schema for the XML format, the sample JSON for the JSON format, or the sample .avsc file for the Avro format.
Column Mapping	The mapping of source data to the data object. Click **View** to see the mapping.
Project Column as Complex Data Type	Project columns as complex data type for hierarchical data. For more information, see the *Data Engineering Integration User Guide*.

Microsoft Azure Data Lake Storage Gen1 Data Object

A Microsoft Azure Data Lake Storage Gen1 data object is a physical data object that represents data in a Microsoft Azure Data Lake Storage Gen1 table. After you create an Azure Data Lake Storage Gen1 connection, create a Microsoft Azure Data Lake Storage Gen1 data object write operation to write to a Microsoft Azure Data Lake Storage Gen1 table.

You can use Microsoft Azure Data Lake Storage Gen1 to store data irrespective of size, structure, and format. Use Microsoft Azure Data Lake Storage Gen1 to process large volumes of data to achieve faster business outcomes.

When you configure the data operation properties, specify the format in which the data object writes data. You can specify XML, JSON, or Avro as format. When you specify XML format, you must provide an XSD file. When you specify Avro format, provide a sample Avro schema in a .avsc file. When you specify JSON, you must provide a sample file.

You cannot run a mapping with a Microsoft Azure Data Lake Storage Gen1 data object on a MapR distribution.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Microsoft Azure Data Lake Storage Gen1 Data Object Properties

The Microsoft Azure Data Lake Storage Gen1 Overview view displays general information about the Microsoft Azure Data Lake Storage Gen1 data object and the object properties that apply to the Microsoft Azure Data Lake Storage Gen1 table you import.

General Properties

You can configure the following properties for a Microsoft Azure Data Lake Storage Gen1 data object:

- Name. Name of the Microsoft Azure Data Lake Storage Gen1 data object.
- Description. Description of the Microsoft Azure Data Lake Storage Gen1 data object.
- Connection. Name of the Microsoft Azure Data Lake Storage Gen1 connection.

Microsoft Azure Data Lake Storage Gen1 Data Object Write Operation Properties

The Data Integration Service writes data to a Microsoft Azure Data Lake Storage Gen1 object based on the data object write operation. The Developer tool displays the data object write operation properties for the Microsoft Azure Data Lake Storage Gen1 data object in the Data Object Operation section.

You can view the data object write operation from the Input and Target properties.

Input properties

Represent data that the Data Integration Service reads from a Microsoft Azure Data Lake Storage Gen1 directory server. Select the input properties to edit the port properties and specify the advanced properties of the data object write operation.

Target properties

Represent data that the Data Integration Service writes to Microsoft Azure Data Lake Storage Gen1. Select the target properties to view data, such as the name and description of the Microsoft Azure Data Lake Storage Gen1 object.

Input Properties of the Data Object Write Operation

Input properties represent data that the Data Integration Service writes to a Microsoft Azure Data Lake Storage Gen1 directory server. Select the input properties to edit the port properties of the data object write operation. You can also specify advanced data object write operation properties to write data to Microsoft Azure Data Lake Storage Gen1 objects.

The input properties of the data object write operation include general properties that apply to the data object write operation. Input properties also include port, source, and advanced properties that apply to the data object write operation.

You can view and change the input properties of the data object write operation from the **General**, **Ports**, **Targets**, **run-time**, and **Advanced** tabs.

Ports Properties - Input Write

The input ports properties list the data types, precision, and scale of the data object write operation.

The following table describes the input ports properties that you must configure in the data object write operation:

Property	Description
Name	The name of the port.
Type	The data type of the port.
Precision	The maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the port.

Run-time Properties

The run-time properties display the name of the connection used for write transformation.

The following table describes the run-time properties that you configure for a Microsoft Azure Data Lake Storage Gen1 write operation:

Property	Description
Connection	Name of the Microsoft Azure Data Lake Storage Gen1 connection.

Advanced Properties

You can use the advanced properties to specify data object write operation properties to write data to a Microsoft Azure Data Lake Store server.

The following table describes the advanced properties that you configure for a Microsoft Azure Data Lake Store write operation:

Property	Description
Tracing Level	By default, the tracing level for every transformation is Normal. Change the tracing level to a Verbose setting when you need to troubleshoot a transformation that is not behaving as expected. Set the tracing level to Terse when you want the minimum amount of detail to appear in the log.
Directory path override	Overrides the default directory path. **Note:** You must manually create a temporary active directory within the target file directory of Azure Data Lake Store connection. The temporary active directory contains the target files to which the data is currently written. After the target file reaches the specified rollover limit, the target file is closed and moved to the specified target file directory from the temporary active directory.
File name override	Overrides the default file name.
Compress newly created File	Not applicable for streaming mappings.
Maintain row order	Not applicable for streaming mappings.
If file exists	Not applicable for streaming mappings.
Interim Directory	The active directory location of the complex file target. This directory stages all the files currently in open state. When the stream rollover condition is met, the files are moved from the interim directory to the target directory.

Column Projection Properties

The Developer tool displays the column projection properties in the **Properties** view of the write operation.

To specify column projection properties, double click on the write operation and select the data object. The following table describes the columns projection properties that you configure for Azure Data Lake Store targets:

Property	Description
Column Name	The field in the target that the data object writes to. This property is read-only.
Type	The native data type of the target. This property is read-only.
Enable Column Projection	Indicates that you use a schema to publish the data to the target. By default, the data is streamed in binary format. To change the streaming format, select this option and specify the schema format.

Property	Description
Schema Format	The format in which you stream data to the target. You can select one of the following formats: - XML - JSON - Avro
Schema	Specify the XSD schema for the XML format, asample file for JSON, or .avsc file for Avro format.
Column Mapping	The mapping of data object to the target. Click **View** to see the mapping.
Project as Hierarchical Type	Project columns as complex data type for hierarchical data. For more information, see the *Data Engineering Integration User Guide*.

Microsoft Azure Data Lake Storage Gen2 Data Object

A Microsoft Azure Data Lake Storage Gen2 data object is a physical data object that represents data in a Microsoft Azure Data Lake Gen2 Storage Gen2. After you create a Microsoft Azure Data Lake Storage Gen2 connection, create a Microsoft Azure Data Lake Storage Gen2 data object write operation to write to a Microsoft Azure Data Lake Gen2 Storage.

Azure Data Lake Storage Gen2 inherits capabilities of Azure Data Lake Storage Gen1, such as file system semantics, directory, and file level security and scale. These capabilities are combined with Azure Blob capabilities, such as low-cost, tiered storage, and high availability.

Azure Data Lake Storage Gen2 is built on Azure Blob storage. You can use Azure Databricks version 5.4 or Azure HDInsight 4.0 to access the data stored in Azure Data Lake Storage Gen2.

When you configure the data object operation properties, specify the format in which the data object writes data. You can specify JSON, Parquet, or Avro as format. When you specify JSON format, you must provide a sample file. When you specify Avro format, you must provide a sample Avro schema in a .avsc file.

You can pass any payload format directly from source to target in Streaming mappings. You can project columns in binary format pass a payload from source to target in its original form or to pass a payload format that is not supported.

Streaming mappings can read, process, and write hierarchical data. You can use array, struct, and map complex data types to process the hierarchical data. You assign complex data types to ports in a mapping to flow hierarchical data. Ports that flow hierarchical data are called complex ports.

For more information about processing hierarchical data, see the *Data Engineering Integration User Guide*.

Microsoft Azure Data Lake Storage Gen2 Data Object Properties

The Azure Data Lake Storage Gen2 **Overview** view displays general information about the Microsoft Azure Data Lake Storage Gen2 data object and the object properties that apply to the Microsoft Azure Data Lake Storage Gen2 table you import.

General Properties

The following table describes the general properties that you configure for Microsoft Azure Data Lake Storage Gen2 data objects:

Property	Description
Name	Name of the Microsoft Azure Data Lake Storage Gen2 data object.
Description	Description of the Microsoft Azure Data Lake Storage Gen2 data object.
Connection	Name of the Microsoft Azure Data Lake Storage Gen2 connection.

Objects Properties

The following table describes the objects properties that you configure for Microsoft Azure Data Lake Storage Gen2 data objects:

Property	Description
Name	Name of the Microsoft Azure Data Lake Storage Gen2 data object.
Description	Description of the Microsoft Azure Data Lake Storage Gen2 data object.
Native Name	Native name of the Microsoft Azure Data Lake Storage Gen2 data object.
Path Information	Path of the Microsoft Azure Data Lake Storage Gen2 data object.

Column Properties

The following table describes the column properties that you configure for Microsoft Azure Data Lake Storage Gen2 data objects:

Property	Description
Name	Name of the Microsoft Azure Data Lake Storage Gen2 data object.
Native Name	Native name of the Microsoft Azure Data Lake Storage Gen2 data object.
Type	Native data type of the Microsoft Azure Data Lake Storage Gen2 data object.
Precision	Maximum number of significant digits for numeric data types, or the maximum number of characters for string data types.
Scale	Scale of the data type.
Description	Description of the Microsoft Azure Data Lake Storage Gen2 data object.

Microsoft Azure Data Lake Store Gen2 Data Object Write Operation Properties

The Data Integration Service writes data to a Microsoft Azure Data Lake Storage Gen2 object based on the data object write operation. You can view the write operation properties for Microsoft Azure Data Lake Storage Gen2 data object in the **Data Object Operation** section in the Developer tool.

General Properties

You can view the general properties for Microsoft Azure Data Lake Storage Gen2 targets in the **Write** view in the Developer tool.

The following table describes the general properties that you view for Microsoft Azure Data Lake Storage Gen2 targets:

Property	Description
Name	The name of the Microsoft Azure Data Lake Storage Gen2 target write operation.
Description	The description of the Microsoft Azure Data Lake Storage Gen2 target write operation.

Ports Properties

Ports properties for a Microsoft Azure Data Lake Storage Gen2 data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Microsoft Azure Data Lake Storage Gen2 targets:

Property	Description
Name	The name of the target port.
Type	The native data type of the target port.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the target port.

Targets Properties

The targets properties list the targets of the Microsoft Azure Data Lake Storage Gen2 data object.

The following table describes the targets property that you can configure for Microsoft Azure Data Lake Storage Gen2 targets:

Property	Description
Target	The target which the Microsoft Azure Data Lake Storage Gen2 data object writes to. You can add or remove targets.

Schema Properties

You can view the schema properties in the **Data Object Operation** view of the write operation in the Developer tool.

To specify schema properties, select the data operation and enable the column projection.

The following table describes the schema properties that you configure for Microsoft Azure Data Lake Storage Gen2 targets:

Property	Description
Column Name	The name field that contains data. This property is read-only.
Type	The native data type of the source. This property is read-only.
Enable Column Projection	Indicates that you use a schema format.
Schema Format	The format in which the data is written. Microsoft Azure Data Lake Storage Gen2 supports JSON, Avro, and Parquet format in streaming mappings.
Schema	Specify a sample file for JSON format, a .avsc file for Avro format, or a Parquet format.
Column Mapping	The mapping of target data to the data object. Click **View** to see the mapping.

Run-time Properties

The run-time properties displays the name of the connection.

The following table describes the run-time property that you configure for Microsoft Azure Data Lake Storage Gen2 targets:

Property	Description
Connection	Name of the Microsoft Azure Data Lake Storage Gen2 connection.

Advanced Properties

You can use the advanced properties to specify data object write operation properties to write data to a Microsoft Azure Data Lake Storage Gen2 data object.

The following table describes the advanced properties that you configure for a Microsoft Azure Data Lake Storage Gen2 write operation:

Property	Description
Operation Type	The type of data object operation. This is a read-only property.
Concurrent Threads	The number of threads to use to move data to the staging area in Microsoft Azure Data Lake Storage Gen2. Default is 10.
Filesystem Name Override	Overrides the default file name.
Directory Override	Overrides the default directory path. The Microsoft Azure Data Lake Store Gen2 directory that you use to write data. Default is root directory. The directory path specified at run time overrides the path specified while creating a connection.
File Name Override	Name of the file to which you want to write data. The file specified at run time overrides the file specified while creating a connection.
Write Strategy	Not applicable for streaming mappings.
Block Size	Not applicable for streaming mappings.
Compression Format	Not applicable for streaming mappings.
Timeout Interval	The number of seconds to wait when attempting to connect to the server. A timeout will occur if the connection cannot be established in the specified amount of time.
Stream Rollover Size in GB	Optional. Target file size, in gigabytes (GB), at which to trigger rollover. A value of zero (0) means that the target file does not roll over based on size. Default is 1 GB.
Stream Rollover Time in Hours	Optional. Length of time, in hours, for a target file to roll over. After the time period has elapsed, the target file rolls over. A value of zero (0) means that the target file does not roll over based on time. Default is 1 Hour.
Interim Directory	The active directory location of the complex file target. This directory stages all the files currently in open state. When the stream rollover condition is met, the files are moved from the interim directory to the target directory.

FileName Port in ADLS Gen2

When you create a data object read or write operation for Microsoft Azure Data Lake Store Gen2 (ADLS Gen2) files, the FileName port appears by default.

When the Spark engine writes to ADLS Gen2 files using a FileName port, it uses the following process to write the data:

1. The Data Integration Service creates separate directories for each value in the FileName port and adds the target files within the directories.

2. The file rollover process closes the current file to which data is being written to and creates a new file based on the configured rollover value.
Effective in 10.4.1, you can configure the following rollover parameters at the design-time based on time and size:

- Stream Rollover Time in Hours. Specify the rollover time in hours for a target file when a certain period of time has elapsed.

- Stream Rollover Size in GB. Specify the size in GB for a target file when the target file reaches a certain size.

3. When a target file reaches the configured rollover value, the target file is rolled over and moved to the specified ADLS Gen2 target location.

4. The Spark engine creates sub-directories in the specified ADLS Gen2 target location for each value in the FileName port.

5. The Spark engine moves the rolled over target files to the sub-directories created for each value in the FileName port in the ADLS Gen2 target location.

You can configure both rollover schemes for an ADLS Gen2 target file. The Spark engine rolls over to based on the first event that triggers. For example, if you configure rollover time to 1 hour and rollover size to 1 GB, the target service rolls the file over when the file reaches a size of 1 GB even if the 1 hour period has not elapsed.

Relational Data Objects

Create a relational data object to write to Hive tables or JDBC-compliant database. To write to Hive tables, create a relational data object with a Hive connection. To write to a JDBC-compliant database, create a relational data object with a JDBC connection.

To use the relational data object as a target in streaming mappings, configure the relational data object write operation properties. You can select the mapping environment and run the mappings on the Spark engine of the Hadoop environment.

Hive targets

When you write to a Hive target in a streaming mapping, you write to a Hive table. You can write to the following types of tables:

- Managed or internal tables. When you write to a managed table or an internal table, Hive writes data to the Hive warehouse directory. If you enable the **Truncate target table** property in the Advanced properties while writing to the Hive table, the data in the table is overwritten. If you do not select this property, data is appended.

- External tables. When you write to an external table, you must truncate the target table to overwrite data. You can write to external partitioned tables but you cannot truncate external partitioned tables.

Truncation of tables happens only once in the beginning when you write data.

Note:

- Data Engineering Streaming does not support complex data types for hive targets.

- For Hadoop HDP version 3.1, HDInsight version 4.0, and Cloudera CDP version 7.1, you can write to Hive tables with only the following properties:

  ACID

  Full Transactional

  Optimized Row Columnar (ORC) file format

  Managed Table

JDBC targets

You can include a JDBC-compliant database as a target in an Informatica mapping. Use the JDBC drivers in the JDBC connection to configure the JDBC parameters. JDBC drivers use the values that you configure in the **User Name** and **Password** fields of the JDBC connection. If you want to write to a JDBC target, you must create a target table .

Relational Data Object Overview Properties

The Data Integration Service uses overview properties when it writes data to a relational data object.

The Overview properties include general properties that apply to the relational data object. They also include column properties that apply to the resources in the relational data object.

General Properties

The following table describes the general properties that you configure for relational data objects:

Property	Description
Name	Name of the relational data object.
Description	Description of the relational data object.

Column Properties

The following table describes the column properties that you can view for relational data objects:

Property	Description
Name	Name of the column.
Native Type	Native data type of the column.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Description	Description of the column.
Nullable	Indicates if the column can contain null values.
Primary Key	Identifies a primary key column in a table that you import from a database source.

Advanced Properties

Advanced properties include run-time and other properties that apply to the relational data object. The Developer tool displays advanced properties for relational data object in the **Advanced** view.

Property	Description
Connection	Name of the Hive connection.
Owner	Name of the resource owner. This property is not applicable for Hive sources and targets.
Resource	Name of the resource.
Database Type	Type of the source. This property is read-only.
Resource Type	Type of the resource. This property is read-only.

Relational Data Object Write Operation Properties

The Data Integration Service uses write operation properties to write data to Hive or JDBC-compliant database.

The data object write operation properties include general, ports, run-time, target, and advanced properties.

General Properties

The general properties include the properties for name, description, and metadata synchronization.

The following table describes the general properties that you configure for the relational data object:

Property	Description
Name	The name of the relational data object write operation. You can edit the name in the **Overview** view. When you use the relational file as a source in a mapping, you can edit the name within the mapping.
Description	The description of the relational data object write operation.
When column metadata changes	Indicates whether object metadata is synchronized with the source. Select one of the following options: - Synchronize output ports. The Developer tool reimports the object metadata from the source. - Do not synchronize. Object metadata may vary from the source.

Data Object Properties

On the Data Object tab, you can specify or change the target, and make relational and customized data object targets dynamic.

The following table describes the data object properties that you configure for Hive targets in a streaming mapping:

Property	Description
Specify By	To specify target columns and metadata, select one of the following options: - Value. The write operation uses the associated data object to specify target columns and metadata. - Parameter. The write operation uses a parameter to specify target columns and metadata. Default is the Value option.
Data Object	If you created the target from an existing data object, the field displays the name of the object. Click Browse to change the data object to associate with the data object.
Parameter	Choose or create a parameter to associate with the target.
At runtime, get data object columns from data source	When you enable this option, the Data Integration Service fetches metadata and data definition changes from target tables to the data object.

Ports Properties

Ports properties include column names and column attributes such as data type and precision.

The following table describes the ports properties that you configure for relational targets:

Property	Description
Name	The name of the column.
Type	The native data type of the column.
Precision	The maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the column.
Column	The name of the column in the resource.
Resource	The name of the resource.

Run-time Properties

The following table describes the run-time properties that you configure for Hive targets:

Property	Description
Connection	Name of the Hive connection.
Owner	Name of the Hive database.
Resource	Name of the resource.

Advanced Properties

The following table describes the advanced properties that you configure for Hive targets in streaming mappings:

Property	Description
Tracing level	Controls the amount of detail in the mapping log file.
Target Schema Strategy	Type of target schema strategy for the target table. You can select one of the following target schema strategies: - RETAIN. The Data Integration Service retains the existing target schema. - CREATE. The Data Integration Service drops the target table at run time and replaces it with a table based on a target table that you identify. - APPLYNEWCOLUMNS. The Data Integration Service alters the target table by applying new columns from the associated data object or mapping flow to the target table. - APPLYNEWSCHEMA. The Data Integration Service alters the target table and applies the new schema from the associated data object or mapping flow to the target table. - FAIL. The Data Integration Service fails the mapping if the target schema from the mapping flow is different from the schema of the target table. - Assign Parameter. You can assign a parameter to represent the value for the target schema strategy and then change the parameter at run time.
DDL Query	The DDL query based on which the Data Integration Service creates or replace the target table at run time. You cannot create a custom DDL query that creates or replaces a Hive table at run time in streaming mappings.
Truncate target table	Truncates the target before loading data. **Note:** If the mapping target is a Hive partition table, you can choose to truncate the target table only with Hive version 0.11.
Truncate Hive Target Partition	Not applicable for streaming mappings.
PreSQL	The SQL command the Data Integration Service runs against the target database before it reads the source. The Developer tool does not validate the SQL.

Property	Description
PostSQL	Not applicable for streaming mappings.
Maintain Row Order	Maintain row order while writing data to the target. Select this option if the Data Integration Service should not perform any optimization that can change the row order.
	When the Data Integration Service performs optimizations, it might lose the row order that was established earlier in the mapping. You can establish row order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When you configure a target to maintain row order, the Data Integration Service does not perform optimizations for the target.

Snowflake Data Objects

A Snowflake data object is a physical data object that represents data based on a Snowflake resource. After you create a Snowflake connection, create a Snowflake data object to write data to Snowflake.

You can configure the data object write operation properties that determine how to write data to Snowflake. After you create a Snowflake data object, create a write operation. You can use the Snowflake data object write operation as a target in streaming mappings.

To run streaming mappings with Snowflake as the target, you must specify the private key to authenticate to Snowflake and the Snowflake internal stage in the advanced properties of the Snowflake data object write operation.

When you run streaming mappings to write to Snowflake, the Spark engine writes data to a streaming Snowflake data frame. The data frame in turn writes to the Snowflake internal stage. After the data is available in the Snowflake internal stage, Snowpipe, which is Snowflake's continuous data ingestion service, loads data from the internal stage to Snowflake.

To avoid conflicts while loading data, do not use the stage used by PowerExchange for Snowflake with any other streaming job or to load other files. Do not manually add, modify, or remove files from the internal stage during streaming.

Snowflake Data Object Write Operation Properties

The Data Integration Service uses write operation properties when it writes data to Snowflake.

General Properties

The Developer tool displays general properties for Snowflake targets in the **Write** view.

The following table describes the general properties for the Snowflake data object write operation:

Property	Description
Name	The name of the Snowflake data object write operation. You can edit the name in the **Overview** view. When you use Snowfalke as a target in a mapping, you can edit the name in the mapping.
Description	The description of the Snowflake data object write operation.

Port Properties

Ports properties for a Snowflake data object include port names and port attributes such as data type and precision.

The following table describes the ports properties that you configure for Snowflake targets:

Property	Description
Name	The name of the target port.
Type	The native data type of the target port.
Precision	The maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.
Scale	The scale of the data type.
Description	The description of the target port.

Run-time Properties

The run-time properties display the name of the connection.

The following table describes the run-time property that you configure for Snowflake targets:

Property	Description
Connection	Name of the Snowflake connection.
Reject file directory	Directory where the reject file exists.
Reject file name	File name of the reject file.

Advanced Properties

Snowflake data object write operation properties include run-time properties that apply to the Snowflake data object.

The Developer tool displays advanced properties for the Snowflake data object operation in the Advanced view.

The following table describes the Advanced properties for a Snowflake data object write operation:

Property	Description
UpdateMode	Not applicable for streaming mappings.
Database	Overrides the database name specified in the connection.
Schema	Overrides the schema name specified in the connection.
Warehouse	Overrides the Snowflake warehouse name specified in the connection.
Role	Overrides the Snowflake user role specified in the connection.
Pre SQL	Not applicable for streaming mappings.
Post SQL	Not applicable for streaming mappings.
Batch Row Size	Not applicable for streaming mappings.
Number of local staging files	Not applicable for streaming mappings.
Truncate Target Table	Truncates the database target table before inserting new rows. Select one of the following options: - True. Truncates the target table before inserting all rows. - False. Inserts new rows without truncating the target table. Default is false.

Property	Description
Additional Write Runtime Parameters	Specify additional run-time parameters. Separate multiple runtime parameters with &. To run streaming mappings to a Snowflake target, configure the following properties: - **pem_private_key**. The private key in PEM format for key pair authentication to access Snowflake. - **streaming_stage**. The internal stage created in Snowflake from which Snowpipe reads the data files. When you specify an internal stage name and the internal stage does not already exist, PowerExchange for Snowflake creates the stage with the name you specify. You must specify a unique name for the streaming stage. Ensure that at a time only one pipe reads from the internal stage and only one Snowflake streaming mapping writes to the internal stage. For example, consider the following run-time parameters: Specify the **pem_private_key** and the **streaming_stage** properties in the following format: `pem_private_key=<PEM_key_value>&streaming_stage=<stage_name>` The **pem_private_key** and **streaming_stage** properties are applicable for streaming mappings on the Spark engine.
Table Name	Overrides the table name of the Snowflake target table.
Rejected File Path	Not applicable for streaming mappings.
Target Schema Strategy	Target schema strategy for the Snowflake target table. You can select one of the following target schema strategies: - RETAIN - Retain an existing target schema. - CREATE - Create a target if it does not exist. - **Note:** To use the CREATE option, you must provide the value of the database and schema name property in the Snowflake connection. - Assign Parameter.

CHAPTER 5

Streaming Mappings

This chapter includes the following topics:

Streaming Mappings Overview

Use the Developer tool to create and run streaming mappings in either Databricks or Hadoop run-time environment and process data that is in JSON, XML, CSV, or Avro format.

Develop a mapping to read, transform, and write data according to your business needs. When you create a streaming mapping, select the environment and the run-time engine. When you run a streaming mapping, the Data Integration Service pushes the processing to nodes on a Databricks Spark engine in the Databricks cluster or on a Spark engine in the Hadoop cluster.

Use the following steps as a guideline when you develop a streaming mapping:

1. Create connections that you want to use to access streaming data.
2. Create input, output, and reusable objects that you want to use in the mapping. Create physical data objects to use as mapping input or output.

3. Create reusable transformations that you want to use.

4. Create rules.

5. Create the streaming mapping.

6. Add objects to the mapping.

7. Link ports between mapping objects to create a flow of data from sources to targets, through transformations.

8. Configure the mapping

9. Validate the mapping to identify errors.

10. Save the mapping and run it to see the mapping output.

Data Objects

Based on the type of source you are reading from or target you are writing to you can create physical data objects to access the different types of data.

The following table lists the data objects that you can include in streaming mappings and the read and write data object operations for each:

Data Object	Source	Data Object Operation	Target	Data Object Operation
Amazon Kinesis	Amazon Kinesis Stream	Read	Amazon Kinesis Firehose Delivery Stream	Write
Amazon S3	-	-	Amazon S3	Write
Azure Data Lake Store	-	-	Microsoft Azure Data Lake Store	Write
Azure Event Hub	Azure Event Hub Service	Read	Azure Event Hub Service	Write
Complex File	-	-	HDFS sequence file or binary file	Write
HBase	-	-	HBase tables	Write
JMS	JMS server	Read	JMS server	Write
Kafka	Kafka broker	Read	Kafka broker	Write
MapR Streams	MapR Stream	Read	MapR Stream	Write
Relational	-	-	Hive table JDBC-compliant database	Write

Creating a Data Object

Create a data object to add to a streaming mapping.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Data Object**.
3. Select the data object that you want to add t the streaming mapping and click **Next**.
   The data object dialog box appears.
4. Enter a name for the data object.
5. Click **Browse** next to the **Location** option and select the target project or folder.
6. Click **Browse** next to the **Connection** option and select the connection that you want to use
7. To add a resource, click **Add** next to the **Selected Resources** option.
   The **Add Resource** dialog box appears.
8. Select the check box next to the resource you want to add and click **OK**.
9. Click **Finish**.
   The data object appears under Data Objects in the project or folder in the **Object Explorer** view.

RELATED TOPICS:

- "Relational Data Objects" on page 128

Creating a Data Object Operation

You can create the data object read or write operation for a data object depending on the source that you read from or target that you write to. You can then add the object operation to a streaming mapping.

1. Select the data object in the **Object Explorer** view.
2. Right-click and select **New** > **Data Object Operation**.

    The **Data Object Operation** dialog box appears.
3. Enter a name for the data object operation.
4. Select the type of data object operation. You can choose to create a StreamRead or StreamWrite operation.

    The Data Integration Service uses read operation properties when it reads data and write operation properties when it writes data.
5. Click **Add**.

    The **Select Resources** dialog box appears.
6. Select the data object for which you want to create the data object operation and click **OK**.
7. Click **Finish**.

The Developer tool creates the data object operation for the selected data object.

Related Topics:

- "Amazon Kinesis Data Object Read Operation Properties" on page 28
- "Amazon Kinesis Data Object Write Operation Properties" on page 70
- "Amazon S3 Data Object Write Operation Properties" on page 73
- "Azure Event Hubs Data Object Read Operation Properties" on page 33
- "Azure Event Hubs Data Object Write Operations" on page 78
- "JMS Data Object Read Operation Properties" on page 49
- "JMS Data Object Write Operation Properties" on page 106
- "Kafka Data Object Read Operation Properties" on page 55
- "Kafka Data Object Write Operation Properties" on page 110
- "MapR Streams Data Object Read Operation Properties" on page 62
- "MapR Streams Data Object Write Operation Properties" on page 118
- "Complex File Data Object Write Operation Properties" on page 87
- "HBase Data Object Write Operation Properties" on page 102
- "Microsoft Azure Data Lake Storage Gen1 Data Object Write Operation Properties" on page 120
- "Relational Data Object Write Operation Properties" on page 130

Transformations in a Streaming Mapping

When you create a streaming mapping, certain transformations are valid or are valid with restrictions.

The transformation types that you can add to a streaming mapping correlate to the types that you can add to a batch mapping with Data Engineering Integration. Most of the restrictions related to batch mappings also apply to streaming mappings. The transformation restrictions mentioned in this guide apply specifically to streaming mappings.

The following table lists the supported transformations in a streaming mapping in the Hadoop and Databricks environment:

Transformations	Hadoop Environment	Databricks Environment
Address Validator transformation	Yes	Yes
Aggregator transformation	Yes	Yes
Classifier transformation	Yes	Yes
Data Masking transformation	Yes	-
Expression transformation	Yes	Yes
Filter transformation	Yes	Yes
Java transformation	Yes	-
Joiner transformation	Yes	Yes
Lookup transformation	Yes	-
Macro transformation	Yes	Yes
Normalizer transformation	Yes	Yes
Parser transformation	Yes	Yes
Python transformation	Yes	Yes
Rank transformation	Yes	Yes
Router transformation	Yes	Yes
Sorter transformation	Yes	-
Standardizer transformation	Yes	Yes
Union transformation	Yes	Yes
Window transformation	Yes	Yes

Note: The Window transformation is not supported for batch mappings.

For information about restrictions for batch mappings, see the *Data Engineering Integration User Guide*.

Address Validator Transformation in a Streaming Mapping

Streaming mappings have the same processing rules as batch mappings on the Spark engine.

Aggregator Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

Mapping Validation

Mapping validation fails in the following situations:

- A streaming pipeline contains more than one Aggregator transformation.
- A streaming pipeline contains an Aggregator transformation and a Rank transformation.
- An Aggregator transformation is upstream from a Lookup transformation.
- An Aggregator transformation is in the same streaming pipeline as a passive Lookup transformation configured with an inequality lookup condition.

Classifier Transformation in a Streaming Mapping

The Classifier transformation is supported without restrictions.

Data Masking Transformation in a Streaming Mapping

Streaming mappings have the same processing rules as batch mappings on the Spark engine.

Expression Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

You cannot use the EXTRACT_STRUCT function for streaming mappings.

Filter Transformation in a Streaming Mapping

Streaming mappings have the same processing rules as batch mappings on the Spark engine.

Java Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

The Data Integration Service ignores the following properties:

- Partitionable. The Data Integration Service ignores the property and can process the transformation with multiple threads.
- Stateless. The Data Integration Service ignores the property and maintains the row order of the input data.

Joiner Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

Mapping Validation

Mapping validation fails in the following situations:

- A Joiner transformation is downstream from an Aggregator transformation.
- A Joiner transformation is downstream from a Rank transformation.
- A streaming pipeline contains more than one Joiner transformation.
- A Joiner transformation joins data from streaming and non-streaming pipelines.

General Guidelines

To specify a join condition, select the TIME_RANGE function from the Advanced condition type on the Join tab and enter a join condition expression. The TIME_RANGE function determines the time range for the streaming events to be joined.

Lookup Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

Mapping Validation

Mapping validation fails in the following situations:

- The lookup is a data object.
- An Aggregator transformation is in the same streaming pipeline as a passive Lookup transformation configured with an inequality lookup condition.
- A Rank transformation is in the same streaming pipeline as a passive Lookup transformation configured with an inequality lookup condition.
- A pipeline contains more than one passive Lookup transformation configured with an inequality condition.

The mapping fails in the following situations:

- The transformation is unconnected.

General Guidelines

Consider the following general guidelines:

- Using a float data type to look up data can return unexpected results.
- Use a Lookup transformation to look up data in a flat file, HDFS, Hive, relational, JDBC V2, and HBase data.
- To avoid cross join of DataFrames, configure the Lookup transformation to ignore null values that match.

HBase Lookups

To use a Lookup transformation on uncached HBase tables, perform the following steps:

1. Create an HBase data object. When you add an HBase table as the resource for a HBase data object, include the ROW ID column.
2. Create a HBase read data operation and import it into the streaming mapping.
3. When you import the data operation to the mapping, select the **Lookup** option.

4. On the Lookup tab, configure the following options:

   - Lookup column. Specify an equality condition on ROW ID

   - Operator. Specify =

5. Verify that format for any date value in the HBase tables is of a valid Java date format. Specify this format in the **Date Time Format** property of the **Advanced Properties** tab of the data object read operation.

Note: If an HBase lookup does not result in a match, it generates a row with null values for all columns. You can add a Filter transformation after the Lookup transformation to filter out null rows.

Mapping validation fails in the following situations:

- The condition does not contain a ROW ID.

- The transformation contains an inequality condition.

- The transformation contains multiple conditions.

- An input column is of a date type.

JDBC V2 Lookups

You can add a JDBC V2 data object read operation as a lookup in a streaming mapping. You can run streaming mappings with JDBC V2 lookup in Azure Databricks service in Microsoft Azure cloud services.

Macro Transformation in a Streaming Mapping

The Macro transformation is supported without restrictions.

Normalizer Transformation in a Streaming Mapping

The Normalizer transformation is supported without restrictions.

Parser Transformation in a Streaming Mapping

The Parser transformation is supported without restrictions.

Python Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

Consider the following restrictions:

- When you close a Jep instance, you might not be able to call CPython modules.

- Do not use the names `inRowsList` and `outRowsList` in the Python code. These names are reserved for the Scala code that the Spark engine uses to interact with the Python environment.

Rank Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

Mapping Validation

Mapping validation fails in the following situations:

- A Rank transformation is in the same streaming pipeline as a passive Lookup transformation configured with an inequality lookup condition.
- A Rank transformation is upstream from a Joiner transformation.
- A streaming pipeline contains more than one Rank transformation.
- A streaming pipeline contains an Aggregator transformation and a Rank transformation.

Router Transformation in a Streaming Mapping

Streaming mappings have the same processing rules as batch mappings on the Spark engine.

Sorter Transformation in a Streaming Mapping

Streaming mappings have additional processing rules that do not apply to batch mappings.

Mapping Validation

Mapping validation fails in the following situation:

- The Sorter transformation in a streaming mapping does not have an upstream Aggregator transformation.

General Guidelines

Consider the following general guidelines:

- The mapping runs in complete output mode if it contains a Sorter transformation.
- To maintain global sort order, ensure that the target is single-partitioned. Source can be single- or multi-partitioned.

Standardizer Transformation in a Streaming Mapping

Streaming mappings have the same processing rules as batch mappings on the Spark engine.

Union Transformation in a Streaming Mapping

Streaming mappings have mapping validation restrictions.

Mapping Validation

Mapping validation fails in the following situation:

- The transformation is configured to merge data from streaming and non-streaming pipelines.

Window Transformation in a Streaming Mapping

The Window transformation is not supported to run on the Data Integration Service or in batch Spark mappings. Certain restrictions do apply to the Window transformation.

For more information about the restrictions, see the .

Stateful Computing

Stateful computing involves storing and retrieving state while evaluating expressions in an Expression transformation.

You can use variable port definitions in Expression transformations to perform stateful computing in a streaming mapping.

Use variable port definitions to perform the following tasks:

- Calculate and store the state in stateful variables when an event arrives.

- Use the stored state and input to compute the output.

When you configure the Expression transformation, configure the partition keys on the **Windowing** tab. When you define variable ports in an Expression transformation, you can optionally specify a partition key that uses one or more input ports. The partition keys determine which columns to group the data by while performing stateful computing. The stored state will have one value for every stateful variable for each value of the partition key.

The evaluation of the ports is ordered. The output ports are computed after the variable ports are computed and contain updated values of variables.

When you configure windowing properties, you define a stateful expression in the Expression transformation. Streaming mappings support all the expression transform function except Window functions and aggregate functions.

For more information about guidelines for configuring variable ports, see the *Informatica Developer Transformation Guide*.

Partitioning Configuration

Configure partitioning by specifying one or more columns under partition keys on the **Windowing** tab.

The following image shows the Windowing tab:
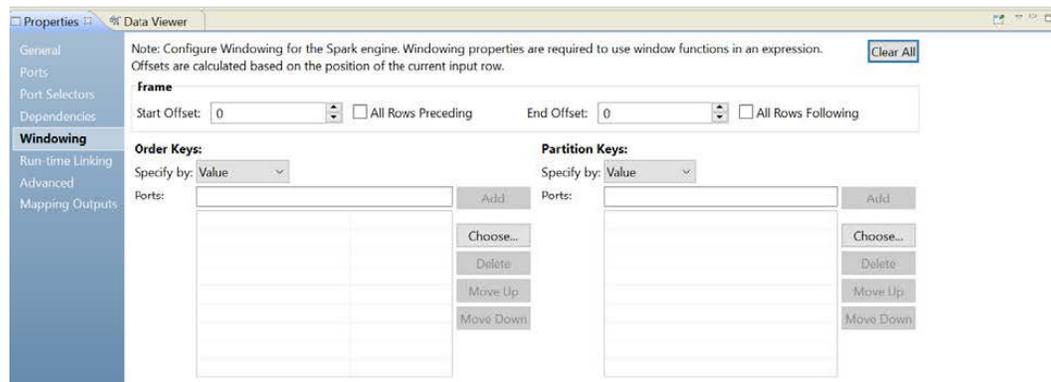
Optionally, configure the partition keys to separate the input rows into different partitions. Configure the partition keys to define partition boundaries, rather than performing the calculation across all inputs. If you do not define partition keys, all the data is included in the same partition. The variable values stored will be global and every row can read and update the same set of variables.

You can specify the partition keys by value or parameter. Select **Value** to use port names.

The following table lists the data type support for variable ports:

Data Type	Initial State Value
String	EMPTY_STRING
Integer	0
Double	0.0
Long	0
Text	EMPTY_STRING
Decimal	0.0

Example

You want to compute average temperature of cities in a region. Add an Expression transformation to your streaming mapping.

Configure the following ports in the transformation:

- city. Input port of string data type.
- temperature. Input port of double data type.
- avg. Output port that displays the average temperature.
- count. Variable port of integer data type with expression 'count+1'.
- average. Variable port of double data type with expression (average * (count-1) + temperature/(count)

You partition the data by "city". The "average" corresponds to previously stored value for the "average" and is computed with each update in the value of "count". Since "count" is already incremented when "average" is evaluated, specify "count-1" in the expression to get the new average.

The following image shows an Expression transformation:

Rules in a Streaming Mapping

A rule expresses the business logic that defines conditions applied to source data. You can add expression rules to streaming mapping to cleanse, change, or validate data. Create expression rules in the Analyst tool.

You might want to use a rule in different circumstances. You can add a rule to cleanse one or more data columns.

Rules that you create in the Analyst tool appear as mapplets in the Developer tool. You can use a mapplet in a mapping or validate the mapplet as a rule.

In a streaming mapping, you can use the following functions in expression rules:

- LastDay
- Add to Date
- Concat
- Date Diff
- Date Time
- Greatest
- Choose
- Least
- Length
- Null
- Reverse
- Truncate
- Convert to Data

For more information about rules, see the *Informatica Profile Guide*.

For more information about mapplets, see the *Developer Mapping Guide*.

Mapping Configurations for Databricks

To configure a streaming mapping for Databricks, configure the connection for the mapping.

The following image shows the validation and execution environments for a Databricks mapping:
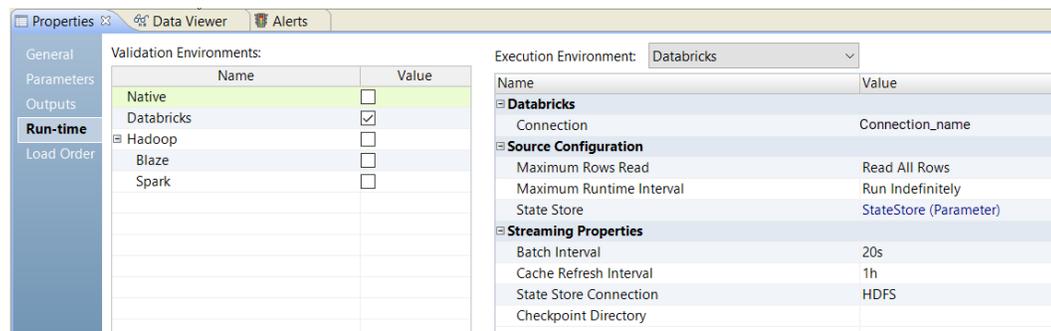
When you configure the mapping, configure the following properties:

Validation environment

The environment that validates the mapping. Select Databricks in the validation environment and select the Databricks engine. The Data Integration Service pushes the mapping logic to the Databricks engine.

Execution environment

The environment that runs the mapping. Select Databricks as the execution environment.

Databricks

Connection: The connection to the Databricks Spark engine used for pushdown of processing. Select **Connection** and browse for a connection or select a connection parameter.

In the case of a mapping failure, to enable the mapping to start reading data from the time of failure, configure the `infaspark.checkpoint.directory` property. For example: `infaspark.checkpoint.directory <directory>`. The directory you specify is created within the directory you specify in the **State Store** property.

Source configuration

Specify the following properties to configure how the data is processed:

- Maximum Rows Read. Specify the maximum number of rows that are read before the mapping stops running. Default is `Read All Rows`.

- Maximum Runtime Interval. Specify the maximum time to run the mapping before it stops. If you set values for this property and the **Maximum Rows Read** property, the mapping stops running after one of the criteria is met. Default is `Run Indefinitely`. A value of `Run Indefinitely` enables the mapping to run without stopping.

- State Store. Specify the DBFS location on the cluster to store information about the state of the Databricks Job. Default is `<Home Directory>/stateStore`
  You can configure the state store as part of the configuration of execution options for the Data Integration Service.

You can use these properties to test the mapping.

Streaming properties

Specify the batch interval streaming properties. The batch interval is number of seconds after which a batch is submitted for processing. Based on the batch interval, the Spark engine processes the streaming data from sources and publishes the data in batches.

Mapping Configurations for Hadoop

To configure a mapping for Hadoop, configure the connection and run-time properties for the mapping.

The following image shows the Hadoop mapping and run-time properties:
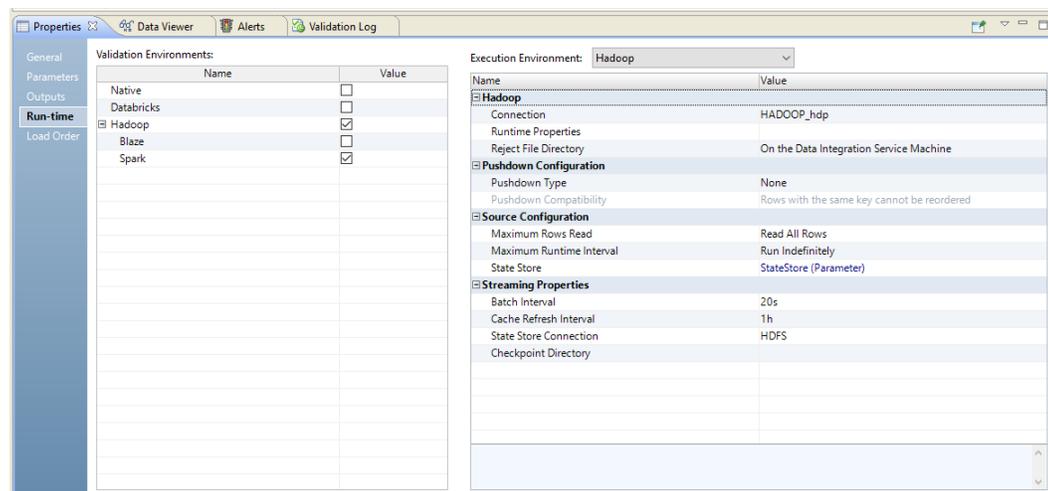
When you configure the mapping, configure the following properties:

Validation environment

> The environment that validates the mapping. Select Hadoop in the validation environment and select the Spark engine. The Data Integration Service pushes the mapping logic to the Spark engine.

Execution environment

> The environment that runs the mapping. Select Hadoop as the execution environment.

Hadoop

> Specify the following properties for the Spark engine:
>
> - Connection. The connection to the Spark engine used for pushdown of processing. Select **Connection** and browse for a connection or select a connection parameter.
>
> - Runtime Properties. An optional list of configuration parameters to apply to the Spark engine. You can change the default Spark configuration properties values, such as `spark.executor.memory`, `spark.sql.shuffle.partitions`, or `spark.driver.cores`.
>   Use the following format:
>
>   `<property1>=<value>`
>
>   - `<property1>` is a Spark configuration property.
>
>   - `<value>` is the value of the property.
>
>   To enter multiple properties, separate each name-value pair with the following text: `&:`
>
>   To use a JMS source or Amazon Kinesis Streams source in the mapping, configure two or more executors for the mapping. For example, use the following configuration:
>
>   `spark.executor.instances=2 &: spark.executor.cores=2 &: spark.driver.cores=1`
>
>   To use an AWS credential profile, configure the following properties for the mapping:
>
>   - `spark.yarn.appMasterEnv.AWS_CREDENTIAL_PROFILES_FILE=<absolute path to the credentials file>/credentials`
>
>   - `spark.executorEnv.AWS_CREDENTIAL_PROFILES_FILE=<absolute path to the credentials file>/credentials`
>
>   - `spark.driverEnv.AWS_CREDENTIAL_PROFILES_FILE=<absolute path to the credentials file>/credentials`
>
>   To maintain global sort order in a streaming mapping that contains a Sorter transformation, you must set the `infaspark.sort.global` property to true and ensure that the Maintain Row Order property is disabled on the target.
>
>   When you use an Amazon S3 target in the mapping to write large amount of data and specify a higher gigabyte value for the `rolloverSize` property, ensure that you add the `spark.hadoop.fs.s3a.connection.maximum` property and set the value of the property to `500` or higher.
>
>   When you use a Google Cloud Storage target in a streaming mapping, you can configure size-based rollover and time-based rollover while creating the mapping.
>
> **Note:** If you enable high-precision in a streaming mapping, the Spark engine runs the mapping in low-precision mode.

Source configuration

> Specify the following properties to configure how the data is processed:
>
> - Maximum Rows Read. Specify the maximum number of rows that are read before the mapping stops running. Default is `Read All Rows`.

- Maximum Runtime Interval. Specify the maximum time to run the mapping before it stops. If you set values for this property and the Maximum Rows Read property, the mapping stops running after one of the criteria is met. Default is `Run Indefinitely`. A value of `Run Indefinitely` enables the mapping to run without stopping.

- State Store. Specify the HDFS location on the cluster to store information about the state of the Spark Job. Default is `<Home Directory>/stateStore`
You can configure the state store as part of the configuration of execution options for the Data Integration Service.

You can use these properties to test the mapping.

Streaming properties

Specify the following streaming properties:

- Batch interval. The Spark engine processes the streaming data from sources and publishes the data in batches. The batch interval is number of seconds after which a batch is submitted for processing.

- Cache refresh interval. You can cache a large lookup source or small lookup tables. When you cache the lookup source, the Data Integration Service queries the lookup cache instead of querying the lookup source for each input row. You can configure the interval for refreshing the cache used in a relational Lookup transformation.
When you run a streaming mapping, if the hive lookup table is in ORC or Parquet format, the minimum value for the cache refresh interval property is 1.

- State Store Connection. You can select an external storage connection for the state store. Default external storage connection is HDFS. You can browse the state store connection property to select Amazon S3, Microsoft Azure Data Lake Stroage Gen1, or Microsoft Azure Data Lake Storage Gen2 as the external storage. You can also have a parameterized connection.

- Checkpoint Directory. You can specify a checkpoint directory to enable a mapping to start reading data from the point of failure when the mapping fails or from the point in which a cluster is deleted. The directory you specify is created within the directory you specify in the State Store property.

Run configurations

The Developer tool applies configuration properties when you run streaming mappings. Set configuration properties for streaming mappings in the **Run** dialog box.

Configure the following source properties:

- Read all rows. Reads all rows from the source.

- Read up to how many rows. The maximum number of rows to read from the source if you do not read all rows.

- Maximum runtime interval. The maximum time to run the mapping before it stops. If you set values for this property and the **Maximum Rows Read** property, the mapping stops running after one of the criteria is met.

When you run the mapping, the Data Integration Service converts the mapping to a Scala program and package it in a JAR file and sends it to the Hadoop cluster. You can view the details in the Spark execution plan in the Developer tool or Administrator tool.

Dynamic Mappings

A dynamic mapping is a mapping that can accommodate changes to sources, targets, and transformation logic at run time. Use a dynamic mapping to manage frequent schema or metadata changes or to reuse the

mapping logic for data sources with different schemas. Configure rules, parameters, and general transformation properties to create a dynamic mapping.

To create a dynamic streaming mapping, enable the refresh schema at the source and target, or enable the mapping flow at the target.

Mapping Flow

You can use the dynamic mapping for a streaming mapping by enabling refresh schema for Confluent Kafka AVRO schema source data objects, if mapping flow is enabled. The mapping flow is enabled for Kafka, Confluent Kafka, and Complex File targets.

Dynamic Port

You can create dynamic ports in a transformation to receive new or changed columns from an upstream transformation. A dynamic port receives one or more columns and generates ports based on input rules. The input rules determine the columns that a dynamic port receives and generates.

Port Mapping

Dynamic port creates a generated port category from an Expression transformation. A static port created in an Expression transformation precedes over generated ports. If a static port and generated port have the same name, the generated port gets renamed. The same is applicable for the header ports. By default, during the runtime, static ports are considered if both static and header port have the same name.

Note: When header and static port names are same, ensure that you link the correct port to the target mapping.

Dynamic Mapping Example

Every week, you receive customer data from different departments that you need to join and aggregate. The departments might periodically change the source schema to include additional columns for departmental analysis. To accommodate the changes to the data source, you create a dynamic mapping. You configure the Read transformation to get data object columns at read time. Create an input rule to include columns that you need and to exclude all other columns.

When your organization wants to manage frequent metadata changes in the streaming data sources, develop a dynamic streaming mapping that can get the metadata changes directly from the data sources at run time.

Confluent Kafka Dynamic Mapping

You can use Confluent Kafka data objects as dynamic sources and targets in a streaming mapping.

Confluent Kafka is a distributed publish-subscribe messaging system that is fast, scalable, and durable. Confluent Kafka topics are partitioned and replicated across multiple nodes thereby allowing distributed processing. You create a Confluent Kafka data object to read data from Kafka brokers or from Confluent Kafka brokers using schema registry.

After you create the Confluent data objects, create a Confluent Kafka streaming mapping to dynamically accommodate changes to source, target, and transformation logic at run time. Configure rules, parameters, and transformation properties to create the dynamic mapping. The dynamic Confluent Kafka streaming mapping manages frequent schema or metadata changes or reuses the mapping logic for data sources with different schemas.

If the data source for a source or target changes, you can configure a mapping to get metadata changes at run time. Configure the Read or Write transformation to accommodate the source or target changes dynamically.

You do not need to manually synchronize the data object and update each transformation before you run the mapping again. The Data Integration Service dynamically determine transformation ports, transformation logic in the ports, and the port links within the mapping.

To enable a streaming mapping to run dynamically, configure it to get data object columns from the data source at run time.

For information about dynamic mappings, see the *Informatica Developer Mapping Guide*.

Example

You run the IT department of a major bank that has millions of customers. You want to monitor network activity in real time. You need to collect network activity data from sources, such as firewalls or network devices to improve security and prevent attacks. The network activity data includes Denial of Service (DoS) attacks and failed login attempts made by customers. The network activity data is written to multiple Confluent Kafka topics with different schema definitions.

You can create one streaming mapping at design time. Configure the streaming mapping to refresh the Confluent Kafka schema registry to adjust the mapping to various sources and targets at run time. The latest schema is fetched from Confluent schema registry and used to synchronize the sources and targets.

Refresh Schema

You can refresh the source or target schema at run time when you enable a streaming mapping to run dynamically. You can refresh the imported metadata before you run the dynamic streaming mapping.

You can enable a streaming mapping to run dynamically using the **At runtime, get data object columns from data source** option in the **Data Object** tab of the Read and Write transformations when you create a mapping.

When you add or override the metadata dynamically, you can include all the existing source and target objects in a single mapping and run the mapping. You do not have to change the source schema to update the data objects and mappings manually to incorporate all the new changes in the mapping.

You can use the mapping template rules to tune the behavior of the execution of such pipeline mapping.

When the Read or Write transformation contains updated ports, such as changes in the port names, data types, precision, or scale, the Data Integration Service fetches the updated ports and runs the mapping dynamically. You must ensure that at least one of the column name in the source or target file is the same as before refreshing the schema to run the dynamic mapping successfully.

Even though the original order of the source or target ports in the table changes, the Data Integration Service displays the original order of the ports in the table when you refresh the schemas at run time.

If there are more columns in the source file as compared to the target file, the Data Integration Service does not map the extra column to the target file and loads null data for all the unmapped columns in the target file.

If the Source transformation contains updated columns that do not match the Target transformation, the Data Integration Service does not link the new ports by default when you refresh the source or target schema. You must create a run-time link between the transformations to link ports at run time based on a parameter or link policy in the **Run-time Linking** tab and update the target schema manually. For information about run-time linking, see the *Informatica Developer Mapping Guide*.

Rules and Guidelines for Dynamic Mapping

Consider the guidelines in this section when you create a dynamic mapping.

Consider the following guidelines for dynamic mappings:

- You can enable the refresh schema only for Confluent Kafka AVRO schema in source data objects.
- XML and binary data format is not supported in the target mapping flow.
- Set the INFA_PARSER_HOME environment variable for AVRO data format in Kafka and Confluent Kafka.

- You must mention the schema registry URL in the connection, if refresh schema is enabled in the source data object.
- For mapping flow, schema parameterization is not certified.
- If mapping flow is enabled, only the latest schema gets refreshed from the schema registry.
- If mapping flow is enabled, create header ports similar to the header names such as partition Id, key, or topic name.

Mapping Validation

When you develop a streaming mapping, you must configure it so that the Data Integration Service can read and process the entire mapping. The Developer tool marks a mapping as not valid when it detects errors that will prevent the Data Integration Service from running the mapping.

The Developer tool considers the following types of validation:

- Environment
- Data object
- Transformation
- Run-time

Environment Validation

The Developer tool performs environment validation each time you validate a streaming mapping.

The Developer tool generates an error in the following scenarios:

- The mapping has Native environment.
- The mapping does not have a Spark validation environment and Hadoop execution environment.
- The mapping has a Hadoop execution environment and has Native, Blaze, and Spark validation environment.

Data Object Validation

When you validate a mapping, the Developer tool verifies the source and target data objects that are part of the streaming mapping.

The Developer tool generates an error in the following scenarios:

- The mapping contains a source or target data object other than the supported data objects.
- The read and write properties of the Kafka, JMS, and complex file data objects are not specified correctly.
- If you add a Kafka or a JMS data object to an LDO mapping, REST mapping, or a mapplet.

Run-time Validation

The Developer Tool performs validations each time you run a streaming mapping.

The Developer tool generates an error in the following scenarios:

- The state store is not configured when you specify the source configuration properties for the mapping.

- Either the Maximum Rows Read or the Maximum Runtime Interval property is not configured when you specify the source configuration properties for the mapping.
- The Maximum Runtime Interval does not have the correct time format.
- The Batch Interval does not have the correct time format.
- If the default value of the Batch Interval and Maximum Runtime Interval properties are not specified.

Transformation Validation

When you validate a streaming mapping, the Developer tool verifies certain transformations as valid or as valid with restrictions.

For example, mapping validation fails if a Joiner transformation joins data from streaming and non-streaming pipelines.

For more information about supported transformations and validation rules in a streaming mapping, see "Transformations in a Streaming Mapping" on page 141.

Mapping Execution Plan

The Data Integration Service generates an execution plan to run mappings on a Spark or Databricks Spark engine.

The Data Integration Service translates the mapping logic into code that the run-time engine can execute. You can view the plan in the Developer tool before you run the mapping and in the Administrator tool after you run the mapping.

For more information, see the *Data Engineering Integration User Guide*.

Monitor Jobs

You can monitor statistics and view log events for a streaming mapping job on the **Monitor** tab of the Administrator tool.

Summary Statistics

Use the **Summary Statistics** view to view graphical summaries of object state and distribution across the Data Integration Services. You can also view graphs of the memory and CPU that the Data Integration Services used to run the objects.

Execution Statistics

Use the **Execution Statistics** view of the **Monitor** tab to monitor properties, run-time statistics, and run-time reports.

The following image shows the monitoring statistics in the **Summary Statistics** tab:
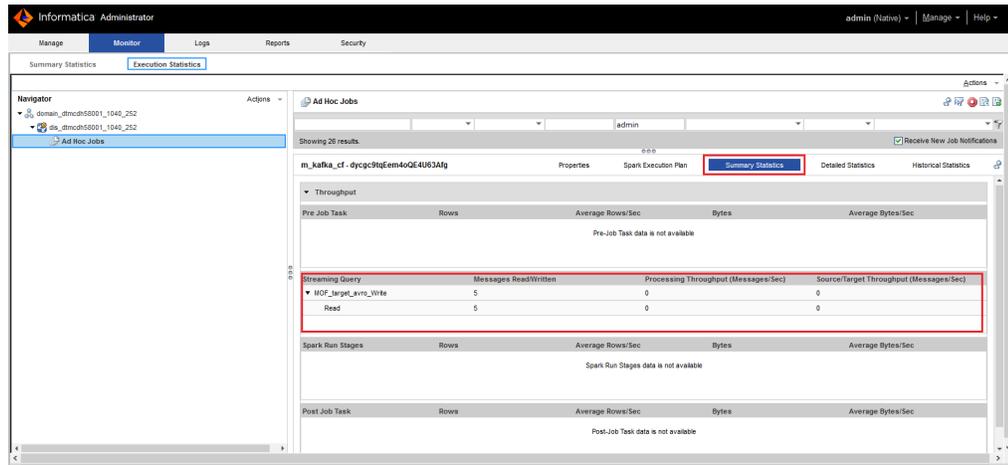
A streaming query has a target and source instance. You can view the number of messages read from source and written to the target streaming mapping. You can also view the throughput of messages processed per second. If a failover occurs, the statistics might not be accurate.

Note: You cannot view the monitoring statistics of the Databricks engine on the Monitor tab.

For more information, see the *Data Engineering Integration User Guide*.

Streaming Mapping Example

You run the IT department of a major bank that has millions of customers. You want to monitor network activity in real time. You need to collect network activity data from various sources such as firewalls or network devices to improve security and prevent attacks. The network activity data includes Denial of Service (DoS) attacks and failed login attempts made by customers. The network activity data is written to Kafka queues.

Create a Streaming mapping to read the network activity data and write the data to HDFS.

In a Hadoop environment, you can use the following objects in the Streaming mapping:

Kafka data object

The input file is a Kafka queue that contains the network activity data.

Create a Kafka data object. Configure a Kafka connection and specify the queue that contains the network activity data as a resource for the data object. Create a data object read operation and configure the properties. Drag the data object into the mapping as a source data object.

Transformations

Add a Lookup transformation to get data from a particular customer ID. Add a Window transformation to accumulate the streamed data into data groups before processing the data.

HDFS complex file data object

Create a complex file data object. Configure an HDFS connection to write to an HDFS sequence file. Create the data object write operation and configure the properties. Drag the data object into the mapping as a target data object.

Link ports between mapping objects to create a flow of data.

The following image shows the sample mapping:
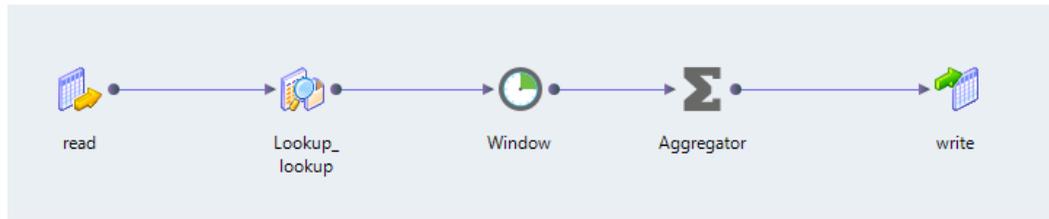
When you run the mapping, the data is read from the Kafka queue and written to the HDFS sequence file.

Data Preview

You can preview data within a mapping in the Developer tool. You can choose sources and transformations in a mapping as preview points. Previewing data helps to design and debug mappings.

You can preview data in streaming mappings configured to run with the following cluster distributions:

- Amazon EMR
- Cloudera CDH
- Cloudera CDP
- Dataproc

The following image shows the run-time properties in the Hadoop execution environment for data preview:
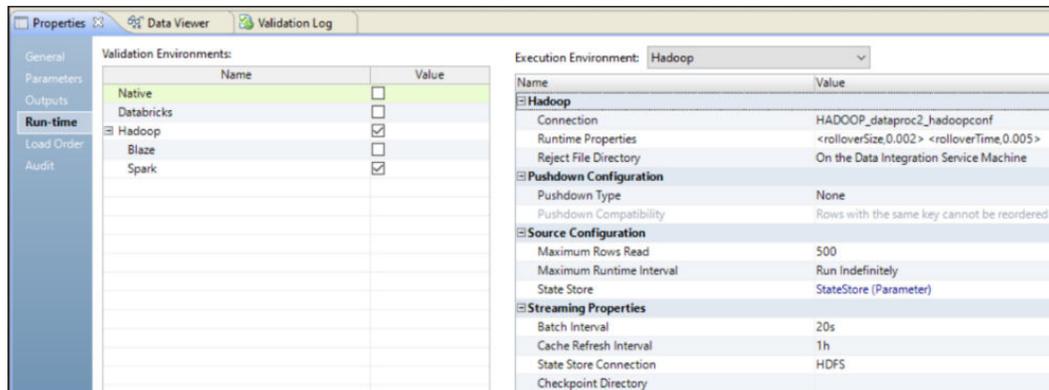
When you configure run-time properties for the Hadoop environment to preview data on streaming jobs in the Data Viewer, consider the following properties:

- You can specify the rollover size or rollover time in the **Execution Environment** area of the Developer tool. The steps to configure the rollover size and rollover time are similar to the configurations when you run a map. The rollover size is the target file size, in gigabytes(GB), at which to trigger rollover. A value of zero (0) means that the target file does not roll over based on size. Default is 100 bytes. The rollover time is the length of time, in hours, for a target file to roll over. After the time period has elapsed, the target file rolls over. A value of zero (0) means that the target file does not roll over based on time. Default is 1 hour.

- You can specify the maximum runtime interval property when you perform data preview on a streaming job in the **Execution Environment** area of the Developer tool. Maximum runtime interval is the maximum time to run the mapping before it stops. Default is 2.5 minutes. If you set values for this property and the **Maximum Rows Read** property, the mapping stops running when one of the criteria is met.

For more information about configuring a streaming mapping, see "Mapping Configurations for Hadoop" on page 149.

Consider the following guidelines when you preview data on any streaming source:

- You cannot preview data on an Aggregator transformation.

- You cannot connect to the downstream transformation when you map the timestamp port in the Windows transformation to the Aggregator transformation.

- When you preview a streaming source that contains a timestamp column, the timestamp column appears blank. However, the data appears in the timestamp column when you export the data.

- You cannot preview data on a Normalizer transformation and a Router transformation with multiple output groups.

For more information about data preview, see the *Data Engineering Integration User Guide*.

Cluster Workflows

You can run a workflow to create an ephemeral cluster that runs mapping and other tasks on a cloud platform cluster.

Create cluster workflows to run on the Amazon EMR or Microsoft Azure HDInsight cloud platforms in the Hadoop environment. Create cluster workflows to run on the Databricks cloud platform in the Databricks environment. The cluster workflow uses other elements that enable communication between the Data Integration Service and the cloud platform, such as a cloud provisioning configuration and a cluster connection.

A cluster workflow contains a Create Cluster task that you configure with information about the cluster to create. If you want to create an ephemeral cluster, you can include a Delete Cluster task. An ephemeral cluster is a cloud platform cluster that you create to run mappings and other tasks, and then terminate when tasks are complete. Create ephemeral clusters to save cloud platform resources.

For more information about cluster workflows, see the *Data Engineering Integration User Guide*.

Ephemeral Cluster in Streaming Mappings

To reduce the cost of resources used to run a cluster, you can run streaming mappings on ephemeral cluster. Create cluster workflow to create ephemeral cluster and delete the cluster at the end of a certain processing period to free up the resources. When the cluster is deleted the information is stored so that it can be used when the cluster starts again.

To resume data process from the point in which a cluster is deleted, you can run streaming mappings on ephemeral cluster by specifying an external storage and a checkpoint directory.

Amazon S3, Microsoft Azure Data Lake Storage Gen1, and Microsoft Azure Data Lake Storage Gen2 can be specified as the external storage in the **State Store Connection** property.

You must also specify a checkpoint directory in the **Checkpoint Directory** property. The checkpoint details will be available on the external storage.

High Availability Configuration

To configure high availability for the streaming mapping, configure a state store directory for the source and guaranteed processing of the messages streamed by the source. Also configure the Spark execution parameters to enable the mapping to run without failing.

To configure high availability, perform the following configurations:

State store configuration

Configure a state store directory. Spark uses the state store directory to store the checkpoint information at regular intervals during the execution of the mapping. If a failure occurs, Spark restarts processing by reading from this state store directory.

Execution parameters

To ensure that the mapping runs without failing, configure the maximum number of tries to submit the mapping to Spark for processing. Configure the `spark.yarn.maxAppAttempts` and `yarn.resourcemanager.am.max-attempts` execution parameters when you configure the mapping properties. The values that you specify for both parameters must be equal and less than the values configured on the CDH or HortonWorks configuration.

Troubleshooting Streaming Mappings

When I run a streaming mapping, the mapping fails, and I see the following errors in the application logs of the Hadoop cluster:

```
User class threw exception: org.apache.spark.SparkException: Job aborted due to stage
failure:
Task 0 in stage 1.0 failed 4 times, most recent failure: Lost task 0.3 in stage 1.0 (TID
4, localhost):
java.lang.Exception: Retry Failed: Total 3 attempts made at interval 10000ms
at
com.informatica.adapter.streaming.hdfs.common.RetryHandler.errorOccured(RetryHandler.java
:74)
at
com.informatica.adapter.streaming.hdfs.HDFSMessageSender.sendMessages(HDFSMessageSender.j
ava:55)
at com.informatica.bootstrap.InfaStreaming$$anonfun$writeToHdfsPathRealtime$1$$anonfun
$apply$5.apply(InfaStreaming.scala:144)
at com.informatica.bootstrap.InfaStreaming$$anonfun$writeToHdfsPathRealtime$1$$anonfun
$apply$5.apply(InfaStreaming.scala:132)
at org.apache.spark.rdd.RDD$$anonfun$foreachPartition$1$$anonfun$apply
$28.apply(RDD.scala:902)
at org.apache.spark.rdd.RDD$$anonfun$foreachPartition$1$$anonfun$apply
$28.apply(RDD.scala:902)
at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1916)
at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1916)
at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
at org.apache.spark.scheduler.Task.run(Task.scala:86)
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

This error occurs if the HDFS NameNode is configured incorrectly.

To resolve this error, ensure that you specify the NameNode URI correctly in the HDFS connection and that the NameNode is up and running.

When I try to run streaming mappings concurrently, a few of the mappings fail and I get the following error in the Data Integration Service logs:

```
Caused by: java.lang.OutOfMemoryError: Java heap space
at java.util.Arrays.copyOf(Arrays.java:3332)
at java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:124)
at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:448)
at java.lang.StringBuilder.append(StringBuilder.java:136)
```

This error occurs when the Data Integration Service does not have sufficient memory to run concurrent mappings. The Data Integration Service logs are located at `<INFA_HOME>/logs/<node name>/services/DataIntegrationService/disLogs/`

To resolve this error, configure the following advanced properties of the Data Integration Service:

- Maximum Heap Size. Specify a minimum value of 2048M. Default is 640M.

- JVM command Line Options. Specify a minimum value of 1024M for the **XX:MaxMetaspaceSize** attribute. Default is 192M.

The streaming mapping execution fails with the following error in the application logs of the Hadoop cluster:

```
Cleaning up the staging area /tmp/hadoop-yarn/staging/cloudqa/.staging/
job_1475754687186_0406
PriviledgedActionException as:cloudqa (auth:PROXY) via yarn (auth:SIMPLE)
cause:org.apache.hadoop.security.AccessControlException:
Permission denied: user=cloudqa, access=EXECUTE, inode="/tmp/hadoop-yarn/
staging":yarn:supergroup:drwx------
at
org.apache.hadoop.hdfs.server.namenode.DefaultAuthorizationProvider.checkFsPermission(Def
aultAuthorizationProvider.java:281)
at
org.apache.hadoop.hdfs.server.namenode.DefaultAuthorizationProvider.check(DefaultAuthoriz
ationProvider.java:262)
at
org.apache.hadoop.hdfs.server.namenode.DefaultAuthorizationProvider.checkTraverse(Default
AuthorizationProvider.java:206)
at
org.apache.hadoop.hdfs.server.namenode.DefaultAuthorizationProvider.checkPermission(Defau
ltAuthorizationProvider.java:158)
at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionCh
ecker.java:152)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:
6621)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:
6603)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkOwner(FSNamesystem.java:6522)
```

This error occurs when a YARN user, Spark engine user, or mapping impersonation user does not have sufficient permission on the `/tmp/hadoop-yarn/staging` directory. Assign required permissions and run the mapping again.

When I run a Streaming mapping that contains an HBase data object, I get the following error:

```
HBaseDataAdapter : java.lang.NullPointerException
at
com.informatica.products.extensions.adapter.hadoop.hive.storagehandler.utils.PwxWriter.cl
ose(PwxWriter.java:165)
at
com.informatica.products.extensions.adapter.hadoop.hive.storagehandler.PwxHiveRecordWrite
r.close(PwxHiveRecordWriter.java:119)
at com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat
$INFAHiveRecordWriter.close(INFAOutputFormat.java:145)
at org.apache.spark.sql.hive.SparkHiveWriterContainer.close(hiveWriterContainers.scala:
109)
```

```
at
org.apache.spark.sql.hive.SparkHiveWriterContainer.writeToFile(hiveWriterContainers.scala
:194)
at org.apache.spark.sql.hive.execution.InsertIntoHiveTable$$anonfun$saveAsHiveFile
$3.apply(InsertIntoHiveTable.scala:131)
at org.apache.spark.sql.hive.execution.InsertIntoHiveTable$$anonfun$saveAsHiveFile
$3.apply(InsertIntoHiveTable.scala:131)
at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
at org.apache.spark.scheduler.Task.run(Task.scala:86)
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor
```

This error occurs when you try to write a null value to a ROW column of an HBase table.

Ensure that you do not write a null value to a ROW column of an HBase table.

When I test a MapR Streams connection, the Developer tool crashes.

This error occurs if you have not completed the required prerequisites.

Ensure that you copy the conf files to the following directory: `<INFA_HOME>\clients\DeveloperClient\hadoop\mapr_5.2.0\conf`

For more information about the prerequisite tasks, see the *Data Engineering Integration Guide*.

When I import a data object with Avro schema in a streaming mapping the mapping fails with the following error:

```
com.informatica.adapter.sdkadapter.exceptions.AdapterSDKException: [SDK_APP_COM_20000]
error [getSchemaConfig()::java.io.IOException: Not a data file.]
```

This occurs when the sample schema file is invalid. When you specify Avro format, provide a valid Avro schema in a .avsc file.

When I run a streaming mapping with an Event Hub source or target, the mapping fails with the following error:

```
com.microsoft.azure.eventhubs.AuthorizationFailedException: Put token failed. status-
code: 401, status-description: InvalidSignature: The token has an invalid signature.
```

This occurs when the Shared Access Policy Name or Shared Access Policy Primary Key is configured incorrectly in the Azure EventHub read data operation, Azure EventHub write data operation, or Azure Eventhub connection. Configure valid values for these properties. When you configure the properties in the Azure EventHub connection, ensure that the policy applies to all data objects that are associated with the connection.

A streaming mapping created in 10.2.0 might not run after upgrading to 10.2.1 when you use JSON schema for column projection for the data you are writing

This might occur if the payload has malformed data.

Remove the malformed data run the mapping.

When I run a streaming mapping on a Cloudera CDH cluster that contains a Kafka source or target, the application does not process the data.

This error occurs when the offset commit topic replication factor configured for the Kafka broker does not match the number of nodes running on the Cloudera CDH cluster.

Perform the following steps to resolve this error:

1. Delete all Kafka topics.

2. Stop the Kafka broker and clear the Kafka log directory specified in the `log.dirs` property of the Kafka broker.

3. Stop ZooKeeper and clear the log directory specified in the `dataDir` property of ZooKeeper.

4. Configure the `Offset Commit Topic Replication Factor` property for the Kafka broker. Specify a value of 1 or 2 for `offsets.topic.replication.factor` property depending on the number of nodes in the cluster.

5. Start ZooKeeper and the Kafka broker.

6. Create the Kafka topics and run the mapping again.

When I run a streaming mapping that contains a JMS target, I get the following error:

```
WebSphere MQ call failed with compcode '2' ('MQCC_FAILED') reason
'2053' ('MQRC_Q_FULL').
at com.ibm.msg.client.wmq.common.internal.Reason.createException
```

This error occurs when the JMS queue that you are writing to is full.

Increase the queue depth of the JMS server and run the mapping again.

In a streaming mapping, when I perform a self-join on source data that has metadata of complex data type, the mapping validation fails at design time.

This error might occur if you have selected the **Sorted Input** property in the advanced properties of the Joiner transformation.

To resolve this error, deselect the **Sorted Input** property and run the mapping again.

When I run a streaming mapping that contains a JMS source, it fails with an `Unexpected JMSMessage payload type` error.

This error might occur in the following situations:

- There is a mismatch between the data type that you write to the queue and the data present in the queue. Clear the JMS queue and then run the mapping.

- There is a mismatch in the data type that you configured for the JMS source and the data type of the streamed data. Verify that the data type that you configure for the source is the same as the data type of the streamed data.

When I edit the schema of a data that is of complex data type, the schema type does not change.

This error occurs because the **Project Column as Complex Data Type** option is not selected.

To resolve this error, when you edit the schema, select the **Project Column as Complex Data Type** option in the columns projection properties of the data object read or write operation properties.

When I run a streaming mapping with a Kafka source and a Hive target, the mapping fails with the following error message:

```
java.io.IOException: Mkdirs failed to create file
```

To resolve this error, set 777 permissions on the hive.exec.scratchdir directory for the user on all the nodes of the cluster, and then run the mapping.

When I run a streaming mapping in AWS Databricks service with a Kafka source and an Amazon S3 target, the mapping returns malformed data and becomes unresponsive. The mapping then stops processing new data.

This error occurs when some of the Databricks nodes do not have access to the Kafka cluster. Configure the Kafka cluster correctly so that it can be accessed by the Databricks cluster.

On AWS Databricks, a streaming mapping fails if the following conditions are true:

- Reads from an Amazon Kinesis source

- Writes to an Amazon S3 target

- Sets the Amazon S3 connection as the state store connection

The mapping fails with the following error message:

```
ERROR Uncaught throwable from user code: com.amazonaws.SdkClientException: Unable to
load AWS credentials from any provider in the chain: [BasicAWSCredentialsProvider:
Access key or secret key is null,
com.amazonaws.auth.InstanceProfileCredentialsProvider@51634e29: The requested metadata
is not found at http://169.254.169.254/latest/meta-data/iam/security-credentials/]
```

The AWS access key ID and AWS secret access key are case-sensitive. For the state store functionality to work, you must add the following access key ID and secret access key to the **Spark Configuration** tab of the Databricks cluster:

spark.hadoop.fs.s3n.awsAccessKeyId

spark.hadoop.fs.s3n.awsSecretAccessKey

CHAPTER 6

Window Transformation

This chapter includes the following topics:

Window Transformation Overview

Use the Window transformation when you want to accumulate streamed data into data groups and then process the data sets. The Window transformation is a passive transformation.

When you read from unbounded sources, you might want to accumulate the data into bounded data groups for further processing. To introduce bounded intervals to unbounded data, use a Window transformation.

When you configure a Window transformation, define the type of window and the data boundaries by time. To specify data boundaries, configure the window size and window slide interval. The window size defines the time interval for which data is accumulated as a data group. The slide interval defines the time interval after which the accumulated data group is processed further. The watermark delay defines the threshold time for a delayed event to be accumulated into a data group.

Window Transformation Types

When you create a Window transformation, you can configure sliding or tumbling windows to specify a time interval for selecting a data group from data streams.

Select one of the following window types when you create a Window transformation:

- Tumbling
- Sliding

When you develop a Window transformation, you need to consider factors, such as the type of window, the window size and window slide interval, and watermark delay that you want to use on the data that the source streams.

Tumbling Window

A tumbling window accumulates data and returns a bounded data group. After the output data is sent, the tumbling window is cleared and a new group of data is accumulated for the next output. Tumbling windows do not overlap. In tumbling windows, the window size and slide interval are the same.

The following image shows a sample 5-second tumbling window:
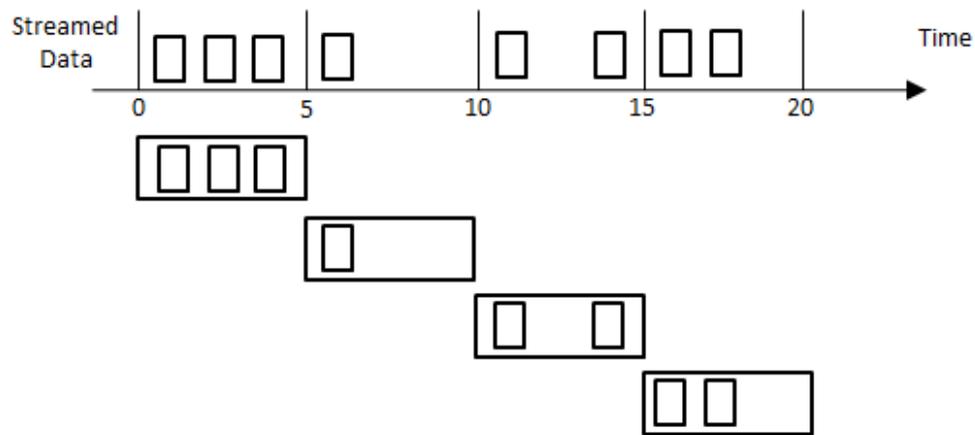
Sliding Window

A sliding window accumulates data and returns a bounded data group. The bounds on the data slide by the time you specify. The data groups that are accumulated can overlap based on the slide interval that you specify.

When you create a sliding Window transformation, the window size must be a multiple of the slide interval.

The following image shows a sample sliding window with a window size of 10 seconds and slide interval of 5 seconds:
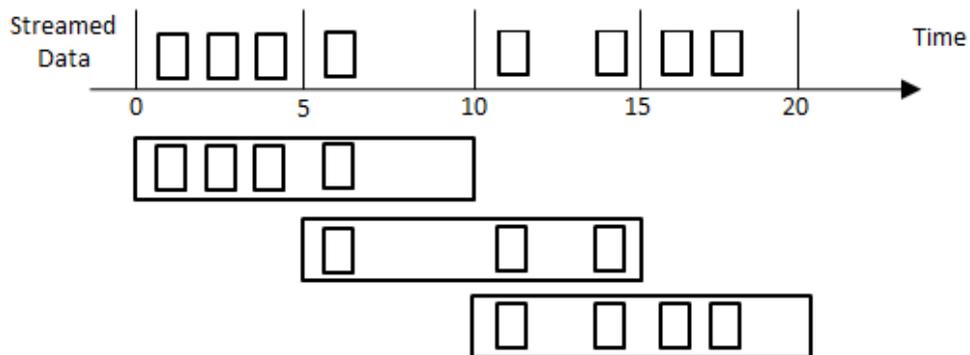
Window Transformation Port Properties

The window port specifies the column that contains the timestamp values based on which you can group the events. The Window Port column is applicable for the ports of date/time data type.

The data accumulated contains the timestamp value at which the event was generated rather than the event arrived at the source. When you create a Window transformation, you can determine the column that you want to use to group the data.

Window Transformation Window Properties

A Window transformation has different window types that allow you to accumulate data groups at different time intervals.

Configure the following window properties for a Window transformation:

Window Type

The type of window transformation you want to create. You can choose tumbling or sliding.

Window Size

The window size defines the time interval for which data is accumulated as a data group. The window size should be a multiple of the batch interval. Specify the window size as a value in units of time or as a parameter of type TimeDuration.

Sliding Interval

The slide interval defines the time interval after which the accumulated data group is processed. Specify the slide interval as a value in units of time or as a parameter of type TimeDuration. Specify the sliding interval if you create a sliding window. By default, the window size and sliding interval are same for tumbling windows.

Watermark Delay

The watermark delay defines threshold time for a delayed event to be accumulated into a data group. Watermark delay is a threshold where you can specify the duration at which late arriving data can be grouped and processed.

If an event data arrives within the threshold time, the data is processed, and the data is accumulated into the corresponding data group. You can specify the watermark delay as a value in units of time or as a parameter of type TimeDuration in the window properties.

The following image shows sample window transformation properties:
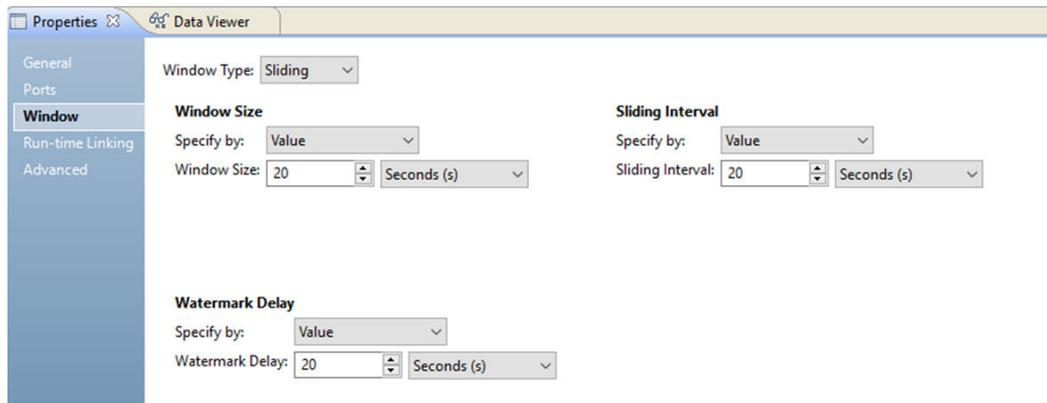
Tumbling Window Transformation Example

You want to calculate the maximum value of a stock price every five minutes for stock prices collected over a five-minute time interval. You can use a tumbling Window transformation.

Create a mapping that reads stock prices and calculates the maximum value every five minute.

The following figure shows the example mapping:

You can use the following objects in your mapping:

Kafka Input

The input, Stock_Read, is a Kafka broker.

Window Transformation

The Window transformation, Window_Tumbling, accumulates data and returns a data group every five minute. Configure a window size of 5 minutes. The default slide interval is 5 minutes. The transformation streams data for five minutes and returns a data group every five minutes.

Aggregator

The Aggregator transformation calculates the maximum value of the stock price.

Kafka Ouptut

The output, Stock_Write, is a Kafka broker.

When you run the mapping, the Data Integration Service reads the data from the Kafka broker and passes it to the Window transformation. The window transformation groups the data and provides a data group every five minutes. The Aggregator transformation provides the maximum stock price. The output is written to a Kafka broker.

Sliding Window Transformation Example

You want to calculate the maximum value of a stock price every minute for stock prices collected over a five-minute time interval. You can use a sliding Window transformation.

Create a mapping that reads stock prices and calculates the maximum value every minute.

The following image shows the example mapping:
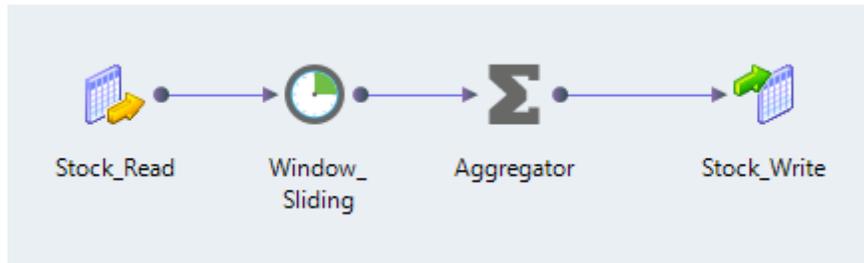
You can use the following objects in your mapping:

Kafka Input

> The input, Stock_Read, is a Kafka broker.

Window Transformation

> The Window transformation, Window_Sliding, accumulates data and returns a data group every minute. Configure a window size of 5 minutes and a slide interval of 1 minute. The transformation streams data for five minutes and returns a data group every minute.

Aggregator

> The Aggregator transformation calculates the maximum value of the stock price.

Kafka Ouptut

> The output, Stock_Write, is a Kafka broker.

When you run the mapping, the Data Integration Service reads the data from the Kafka broker and passes it to the Window transformation. The window transformation groups the data and provides a data group every minute. The Aggregator transformation provides the maximum stock price. The output is written to a Kafka broker.

Rules and Guidelines for the Window Transformations

Certain transformations are valid with restrictions with the Window transformation. The following table describes the rules and guidelines for transformations:

Transformation	Rules and Guidelines
Aggregator	The Aggregator transformation is a multi-group active transformation. The following rules apply to Aggregator transformations: - You must use a Window transformation directly upstream from an Aggregator transformation in a streaming mapping. - If you connect the Window port from a Window transformation to an Aggregator transformation, you cannot connect the Window port to any downstream transformation. - If a mapplet contains an Aggregator transformation, you must include a Window transformation directly upstream from the mapplet. - You cannot perform the group by aggregations on date/time data type port marked as a Window port. If you want to perform aggregations on the date/time data type port, you must create a date/time data type port with the timestamp values, and then perform the group by aggregations on the newly created data type port.
Joiner	The Joiner transformation is a multi-group active transformation. The following rules apply to Joiner transformations: - You must use a Window transformation directly upstream from a Joiner transformation in a streaming mapping. - You must use a Window transformation between the streaming source and any Joiner transformation in a streaming mapping. - The upstream Window transformations in pipelines to a Joiner transformation must have the same slide intervals. - The downstream Window transformations in pipelines to a Joiner transformation must have the same slide intervals. - If you connect the Window port from a Window transformation to a Joiner transformation, you cannot connect the Window port to any downstream transformation. - If a mapplet contains a Joiner transformation, you must include a Window transformation directly upstream from the mapplet.
Lookup	A Lookup transformation does not require a Window transformation between a streaming source and itself.
Rank	A Rank transformation in a streaming mapping or a mapplet must have an upstream Window transformation.
Sorter	The Sorter transformation is an active transformation. The following rules apply to Sorter transformations: - You must use a Window transformation between the streaming source and the Sorter transformation in a streaming mapping. - The Window transformation upstream from an Aggregator transformation will be ignored if the mapping contains a Sorter transformation.

Transformation	Rules and Guidelines
Union	The following rules apply to Union transformations: - A Union transformation does not require a Window transformation between a streaming source and itself. - If one pipeline leading to a Union transformation has a Window transformation, all streaming pipelines must have a Window transformation. All downstream Window transformations in the pipelines leading to the Union transformations must have the same slide intervals.
Window	The following rules apply to Window transformations: - You cannot add a Window transformation to a Logical Data Object mapping or mapplet. - A Window transformation must have at least one upstream streaming source. - All Window transformations must have a slide interval that is a multiple of the mapping batch interval. - A Window transformation that is downstream from another Window transformation must have a slide interval that is a multiple of the slide interval of the upstream Window transformation. - The slide interval of a sliding Window transformation must be less than window size. - The format of the parameter of the window size must have the TimeDuration parameter type. - The window size and the slide interval of a Window transformation must be greater than 0. - The window port from the Window transformation cannot be connected to more than one downstream transformation. - A Window transformation must be added between a streaming source and an Aggregator, or a Joiner transformation. - A Window transformation must not be added to a Logical Data Object mapping, REST mapping, or a mapplet.

APPENDIX A

Connections

This appendix includes the following topics:

Connections Overview

Define the connections that you want to use to access data in Kafka brokers, JMS servers, HDFS files, Hive tables, Amazon Kinesis streams, MapR streams, or HBase resources. You can create the connections using the Developer tool and infacmd.

You can create the following types of connections:
Amazon S3

Create an Amazon S3 connection to write data to Amazon S3.

Databricks

Create a Databricks connection to run mappings in the Databricks environment. For information about Databricks Cloud Provisioning, see the *Data Engineering Integration User Guide*.

Hadoop

Create a Hadoop connection to run mappings on the Hadoop cluster. Select the Hadoop connection if you select the Hadoop run-time environment. You must also select the Hadoop connection to validate a mapping to run on the Hadoop cluster.

For more information about the Hadoop connection properties, see the *Data Engineering Integration User Guide*.

HBase

Create an HBase connection to write data to an HBase resource.

HDFS

Create an HDFS connection to write data to an HDFS binary or sequence file.

Hive

Create a Hive connection to write data to Hive tables.

For more information, see the *Data Engineering Administrator Guide*.

JDBC

Create a JDBC connection when you perform a lookup on a relational database using Sqoop.

For more information about the JDBC connection properties, see the *Data Engineering Integration User Guide*.

Microsoft Azure Data Lake Store

Create a Microsoft Azure Data Lake Store connection to write to a Microsoft Azure Data Lake Store.

Messaging

Create a Messaging connection to access data as it becomes available, and to run a streaming mapping on a Spark engine. You can create the following types of messaging connections:

- Amazon Kinesis. Create an Amazon Kinesis connection to read from Amazon Kinesis Streams or write to Amazon Kinesis Firehose Delivery Streams.
- Azure Event Hub. Create an Azure Event Hub connection to read from or write to Microsoft Event Hubs.
- Confluent Kafka. Create a Confluent Kafka connection to read from or write to a Confluent Kafka broker.
- JMS. Create a JMS connection to read from or write to a JMS server.
- Kafka. Create a Kafka connection to read from or write to a Kafka broker.
- MapR Streams. Create a MapR Streams connection read from or write to MapR Streams.

Amazon Kinesis Connection

The Amazon Kinesis connection is a Messaging connection. Use the Amazon Kinesis connection to access Amazon Kinesis Data Streams as source or Amazon Kinesis Data Firehose as target. You can create and manage an Amazon Kinesis connection in the Developer tool or through infacmd.

When you configure an Amazon Kinesis connection, you configure the following properties:

- Type of service.
- Access keys (access key ID and secret access key) to interact with Amazon Web Services.
- Region that hosts the service that you are trying to use.
- Credential profile to interact with Amazon Web Services.

General Properties

The following table describes the general connection properties for the Amazon Kinesis connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) − + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Messaging/Amazon Kinesis.

Connection Properties

The following table describes the connection properties for the Amazon Kinesis connection:

Property	Description
Service	The type of Kinesis Service that the connection is associated with. Select one of the following service types: - Kinesis Firehose. Select this service to write to Kinesis Firehose Delivery Stream. - Kinesis Streams. Select this service to read from Kinesis Streams
AWS Access Key ID	The access key ID of the Amazon AWS user account.
AWS Secret Access Key	The secret access key for your Amazon AWS user account.

Property	Description
Region	Region where the endpoint for your service is available. You can select one of the following values: - us-east-2. Indicates the US East (Ohio) region. - us-east-1. Indicates the US East (N. Virginia) region. - us-west-1. Indicates the US West (N. California) region. - us-west-2. Indicates the US West (Oregon) region. - ap-northeast-1. Indicates the Asia Pacific (Tokyo) region. - ap-northeast-2. Indicates the Asia Pacific (Seoul) region. - ap-northeast-3. Indicates the Asia Pacific (Osaka-Local) region. - ap-south-1. Indicates the Asia Pacific (Mumbai) region. - ap-southeast-1. Indicates the Asia Pacific (Singapore) region. - ap-southeast-2. Indicates the Asia Pacific (Sydney) region. - ca-central-1. Indicates the Canada (Central) region. - cn-north-1. Indicates the China (Beijing) region. - cn-northwest-1. Indicates the China (Ningxia) region. - eu-central-1. Indicates the EU (Frankfurt) region. - eu-west-1. Indicates the EU (Ireland) region. - eu-west-2. Indicates the EU (London) region. - eu-west-3. Indicates the EU (Paris) region. - sa-east-1. Indicates the South America (São Paulo) region.
Connection Timeout (in msec)	Number of milliseconds that the Integration service waits to establish a connection to the Kinesis Stream or Kinesis Firehose after which it times out.
Authentication Type	The type of authentication. Select one of the following values: - AWS Credential Profile - Cross-account IAM Role The default value is AWS Credential Profile.
AWS Credential Profile Name	Required if you use the AWS credential profile authentication type. An AWS credential profile defined in the credentials file. A mapping accesses the AWS credentials through the profile name at run time. If you do not provide an AWS credential profile name, the mapping uses the access key ID and secret access key that you specify when you create the connection.
ARN of IAM Role	Required if you use the cross-account IAM role authentication type. The Amazon Resource Name specifying the role of an IAM user.
External ID	Required if you use the cross-account IAM role authentication type and if the external ID is defined by the AWS account. The external ID for an IAM role is an additional restriction that you can use in an IAM role trust policy to designate who can assume the IAM role.

Creating an Amazon Kinesis Connection Using infacmd

You can use the infacmd command line program to create an Amazon Kinesis connection.

To create an Amazon Kinesis connection through Kinesis Streams on UNIX using cross-account IAM role, run the following command:

```
infacmd createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct AMAZONKINESIS -o "AWS_ACCESS_KEY_ID=<access key
id> AWS_SECRET_ACCESS_KEY=<secret access key> ConnectionTimeOut=10000 Region=<RegionName>
ServiceType='Kinesis Streams' RoleArn=<ARN of IAM role> ExternalID=<External ID>
AuthenticationType='Cross-account IAM Role'"
```

To create an Amazon Kinesis connection through Kinesis Firehose on UNIX using AWS credential profile, run the following command:

```
infacmd createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct AMAZONKINESIS -o "AWS_ACCESS_KEY_ID=<access key
id> AWS_SECRET_ACCESS_KEY=<secret access key> ConnectionTimeOut=10000 Region=<RegionName>
ServiceType='Kinesis Firehose' Profilename=<AWS credential profile> AuthenticationType='AWS
Credential Profile'"
```

For more information about the CreateConnection command, see the *Informatica Command Reference*.

Amazon S3 Connection

Create an Amazon S3 connection to write data to Amazon S3.

You can create and manage an Amazon S3 connection in the Developer tool or through `infacmd`.

General Properties

The following table describes the general connection properties for the Amazon S3 connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Amazon S3.

Connection Properties

The following table describes the connection properties for the Amazon S3 connection:

Property	Description
Access Key	The access key ID of the Amazon S3 user account.
Secret Key	The secret access key for your Amazon S3 user account.

Property	Description
Folder Path	The complete path to Amazon S3 objects. The path must include the bucket name and any folder name. Do not use a slash at the end of the folder path. For example, `<bucket name>/<my folder name>`.
Master Symmetric Key	Not applicable.
Region Name	Select the AWS region in which the bucket you want to access resides. Select one of the following regions: - Asia Pacific (Mumbai) - Asia Pacific (Seoul) - Asia Pacific (Singapore) - Asia Pacific (Sydney) - Asia Pacific (Tokyo) - AWS GovCloud (US) - Canada (Central) - China (Beijing) - China (Ningxia) - EU (Ireland) - EU (Frankfurt) - EU (London) - EU (Paris) - South America (Sao Paulo) - US East (Ohio) - US East (N. Virginia) - US West (N. California) - US West (Oregon) Default is US East (N. Virginia).
Customer Master Key ID	Optional. Specify the customer master key ID or alias name generated by AWS Key Management Service (AWS KMS). You must generate the customer master key for the same region where Amazon S3 bucket reside. You can specify any of the following values: **Customer generated customer master key** Enables server-side encryption. **Default customer master key** Enables server-side encryption. Only the administrator user of the account can use the default customer master key ID to enable client-side encryption.

For more information about creating an access key an secret key, see AWS documentation.

Creating an Amazon S3 Connection Using infacmd

You can use the infacmd command line program to create an Amazon S3 connection.

To create an Amazon S3 connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct AMAZONS3 -o "AccessKey=<AccessKey>
SecretKey=<SecretKey> FolderPath=<FolderPath> RegionName=<RegionName>"
```

For more information about the CreateConnection command and Amazon S3 connection options, see the *Informatica Command Reference.*

Azure Event Hub Connection

Use the Azure Event Hub connection to access Azure Event Hub as source or target. You can create and manage an Azure Event Hub connection in the Developer tool or through infacmd.

General Properties

The following table describes the general connection properties for the Azure Event Hub connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) − + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Messaging/Azure Event Hub

Connection Properties

The following table describes the connection properties for the Azure Event Hub connection:

Property	Description
Tenant ID	The ID of the tenant that the data belongs to. This ID is the Directory ID of the Azure Active Directory.
Subscription ID	The ID of the Azure subscription.
Resource Group Name	The name of the Azure resource group associated with the event hub namespace.
Client Application ID	The ID of the application created under the Azure Active Directory.
Client Secret Key	The secret key generated for the application.
Event Hub Namespace	The name of the Event Hub Namespace that is associated with the resource group name.

Property	Description
Shared Access Policy Name	The name of the Event Hub Namespace Shared Access Policy. The policy must apply to all data objects that are associated with this connection. To read from Event Hubs, you must have Listen permission. To write to an Event hub, the policy must have Send permission.
Shared Access Policy Primary Key	The primary key of the Event Hub Namespace Shared Access Policy.

Creating an Azure Event Hub Connection Using infacmd

You can use the infacmd command line program to create an Azure Event Hub connection.

To create an Azure Event Hub connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct AZUREEVENTHUB -o tenantID='tenant id'
subscriptionID='subscription id' resourceName='resource name' clientID='client id'
clientSecretKey='secret key' eventHubNamespace='namespace' sasPolicyName=<policy name>
sasPolicyPrimaryKey=<policy key>
```

Cassandra Connection Properties

When you set up a Cassandra connection, you must configure the connection properties.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes the Cassandra connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}]	\ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. The ID must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	Optional. The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Not applicable for Data Engineering Streaming.
Type	The connection type. Select **Cassandra**.

Property	Description
Host Name	Host name or IP address of the Cassandra server.
Port	Cassandra server port number. Default is 9042.
User Name	User name to access the Cassandra server.
Password	Password corresponding to the user name to access the Cassandra server.
Default Keyspace	Name of the Cassandra keyspace to use by default.
SQL Identifier Character	Type of character that the database uses to enclose delimited identifiers in SQL or CQL queries. The available characters depend on the database type. Select **None** if the database uses regular identifiers. When the Data Integration Service generates SQL or CQL queries, the service does not place delimited characters around any identifiers. Select a character if the database uses delimited identifiers. When the Data Integration Service generates SQL or CQL queries, the service encloses delimited identifiers within this character.
SSL Mode	Select **disabled**. Not applicable for PowerExchange for Cassandra JDBC. SSL mode indicates the encryption type to use for the connection. You can choose a mode from the following SSL modes: - Disabled - One way - Two way
SSL Truststore Path	Not applicable for PowerExchange for Cassandra JDBC or when **disabled** SSL mode is selected. Absolute path and file name of the SSL truststore file that contains certificates of the trusted SSL server.
SSL Truststore Password	Not applicable for PowerExchange for Cassandra JDBC or when **disabled** SSL mode is selected. Password for the SSL truststore.
SSL Keystore Path	Not applicable for PowerExchange for Cassandra JDBC or when **disabled** SSL mode is selected. Absolute path and file name of the SSL keystore file that contains private keys and certificates for the SSL server.
SSL Keystore Password	Not applicable for PowerExchange for Cassandra JDBC or when **disabled** SSL mode is selected. Password for the SSL keystore.
Additional Connection Properties	Enter one or more JDBC connection parameters in the following format: `<param1>=<value>;<param2>=<value>;<param3>=<value>` PowerExchange for Cassandra JDBC supports the following JDBC connection parameters: - BinaryColumnLength - DecimalColumnScale - EnableCaseSensitive - EnableNullInsert - EnablePaging - RowsPerPage - StringColumnLength - VTTableNameSeparator

Confluent Kafka Connection

The Confluent Kafka connection is a Messaging connection. Use the Confluent Kafka connection to access a Kafka broker or a Confluent Kafka broker as a source or a target. You can create and manage a Confluent Kafka connection in the Developer tool or through infacmd.

When you configure a Confluent Kafka connection, you configure the following properties:

- List of Kafka brokers or Confluent Kafka brokers that the connection reads from or writes to.
- Number of seconds the Integration Service attempts to reconnect to the database if the connection fails.
- Version of the Confluent Kafka messaging broker.

General Properties

The following table describes the general connection properties for the Confluent Kafka connection:

Property	Description
Name	Name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	Description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	Domain where you want to create the connection. Select the domain name.
Type	Connection type. Select `Messaging/ConfluentKafka.`

Confluent Kafka Broker Properties

The following table describes the Kafka broker properties for the Confluent Kafka connection:

Property	Description
Kafka Broker List	Comma-separated list of Confluent Kafka brokers that maintain the configuration of the Confluent Kafka messaging broker. To specify a Confluent Kafka broker, use the following format: `<IP address>:<port>`
Retry Timeout	Number of seconds after which the Data Integration Service attempts to reconnect to the Confluent Kafka broker to read or write data. If the source or target is not available for the time you specify, the mapping execution stops to avoid any data loss.
Kafka Broker Version	Version of the Confluent Kafka messaging broker.

Property	Description
Additional Connection Properties	Optional. Comma-separated list of connection properties to connect to the Kafka broker.
Schema Registry URL	Location and port of the schema registry provider on which to connect.

Additional Connection Properties

You can use the following syntax for specifying the additional connection properties:

```
request.timeout.ms=<value>,session.timeout.ms=<value>,
fetch.max.wait.ms=<value>,heartbeat.interval.ms=<value>,
security.protocol=SASL_PLAINTEXT,sasl.kerberos.
service.name=<kerberos name>,sasl.mechanism=GSSAPI,
sasl.jaas.config=com.sun.security.auth.module.
Krb5Login Modulerequired useKeyTab=true
doNotPrompt=true storeKey=true client=true
keyTab="<Keytab Location>" principal="<principal>";
```

SSL Properties

The following table describes the SSL properties for the Confluent Kafka connection:

Property	Description
SSL Mode	Optional. SSL mode indicates the encryption type to use for the connection. You can choose one of the following SSL modes: - Disabled - One way - Two way The default value is `Disabled`.
SSL TrustStore File Path	Required when `One way` SSL mode is selected. Absolute path and file name of the SSL truststore file that contains certificates of the trusted SSL server.
SSL TrustStore Password	Required when `One way` SSL mode is selected. Password for the SSL truststore.
SSL KeyStore File Path	Required when `Two way` SSL mode is selected. Absolute path and file name of the SSL keystore file that contains private keys and certificates for the SSL server.
SSL KeyStore Password	Required when `Two way` SSL mode is selected. Password for the SSL keystore.
Additional Security Properties	Optional. Comma-separated list of connection properties to connect to the Confluent Kafka broker in a secured way.

Creating a Confluent Kafka Connection Using infacmd

You can use the infacmd command line program to create a Confluent Kafka connection.

To create a Confluent Kafka connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password>
-cn <connection name> -cid <connection id> -ct ConfluentKafka -o
"kfkBrkList='<host1:port1>,<host2:port2>,<host3:port3>' kafkabrokerversion='<version>'
schemaregistryurl='<schema registry URL>'"
```

For more information about the CreateConnection command, see the *Informatica Command Reference*.

Google Cloud Storage Connection Properties

When you set up a Google Cloud Storage connection, you must configure the connection properties.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes the Google Cloud Storage connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters:~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. The ID must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	Optional. The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select **Google Cloud Storage**.
Project ID	Specifies the project_id value present in the JSON file that you download after you create a service account. If you have created multiple projects with the same service account, enter the ID of the project that contains the bucket that you want to connect to.
Service Account ID	Specifies the client_email value present in the JSON file that you download after you create a service account.
Service Account Key	Specifies the private_key value present in the JSON file that you download after you create a service account.

Google PubSub Connection Properties

When you create a Google PubSub connection, you must configure the connection properties.

The following table describes the Google PubSub connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) – + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Connection Type	The connection type. Select **Pub Sub** connection type.
Client Email	The `client_email` value available in the JSON file that you download after you create a service account.
Client Id	The `client_id` value available in the JSON file that you download after you create a service account.
Private Key Id	The `private_key_id` value available in the JSON file that you download after you create a service account.
Private Key	The `private_key` value available in the JSON file that you download after you create a service account.
Project Id	The `project_id` value available in the JSON file that you download after you create a service account.

HBase Connection

Create an HBase connection to write data to an HBase table.

You can create and manage an HBase connection in the Developer tool or through infacmd.

General Properties

The following table describes the general connection properties for the HBase connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ \` ! $ % ^ & * ( ) − + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select on the folloiwng options: - HBase - MapR-DB

Connection Properties

The following table describes the connection properties for the HBase connection:

Property	Description
Database Type	The connection type. Select on the following options: - HBase - MapR-DB
Cluster Configuration	The name of the cluster configuration associated with the Hadoop environment. For more information about cluster configuration, see the *Data Engineering Integration Administrator Guide*.
MapR-DB Database Path	Database path that contains the MapR-DB table that you want to connect to. Enter a valid MapR cluster path. When you create an HBase data object for MapR-DB, you can browse only tables that exist in the MapR-DB path that you specify in the Database Path field. You cannot access tables that are available in sub-directories in the specified path. For example, if you specify the path as /user/customers/, you can access the tables in the customers directory. However, if the customers directory contains a sub-directory named regions, you cannot access the tables in the following directory: `/user/customers/regions`

Creating an HBASE Connection Using infacmd

You can use the infacmd command line program to create an HBASE connection.

To create an HBASE connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct HBase -o DatabaseType=HBase CLUSTERCONFIGID=ConfId
```

For more information about the CreateConnection command and the HBASE connection options, see the *Informatica Command Reference*.

HDFS Connection

Create an HDFS connection to write data to an HDFS binary or sequence file.

You can create and manage an HDFS connection in the Developer tool or through infacmd.

General Properties

The following table describes the general connection properties for the HDFS connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Hadoop File System.

Connection Properties

The following table describes the connection properties for the HDFS connection:

Property	Description
User Name	User name to access HDFS.
NameNode URI	The URI to access HDFS. Use the following format to specify the NameNode URI in Cloudera and Hortonworks distributions: `hdfs://<namenode>:<port>` Where - `<namenode>` is the host name or IP address of the NameNode. - `<port>` is the port that the NameNode listens for remote procedure calls (RPC). Use the `maprfs:///` format to specify the NameNode URI in MapR distribution.
Cluster Configuration	The name of the cluster configuration associated with the Hadoop environment. For more information about cluster configuration, see the *Data Engineering Integration Administrator Guide*.

Creating an HDFS Connection Using infacmd

You can use the infacmd command line program to create an HDFS connection.

To create an HDFS connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct HadoopFileSystem -o NameNodeURL='hdfs://
<namenode>:<port>' USERNAME='username' clusterConfigId='ConfId'
```

For more information about the CreateConnection command and HDFS connection options, see the *Informatica Command Reference*.

Hive Connection

Create an Hive connection to write data to a Hive table.

You can create and manage a Hive connection in the Developer tool or through infacmd.

Creating a Hive Connection Using infacmd

You can use the infacmd command line program to create a Hive connection.

To create a Hive connection on UNIX, run the following command:

```
sh infacmd.sh CreateConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct HIVE -o "CLUSTERCONFIGID='ConfId'
metaDataConnString='jdbc:hive2://<hostname>:<port>/<db>' enableQuotes='false'
HIVEWAREHOUSEDIRECTORYONHDFS='/user/hive/warehouse' DATABASENAME='default'
BYPASSHIVEJDBCSERVER='false' ConnectString='jdbc:hive2://<hostname>:<port>/<db>
databaseName=default defaultFSURI= hdfs://<node name>:<port>'"
```

For more information about the CreateConnection command and Hive connection options, see the *Informatica Command Reference.*

JMS Connection

The JMS connection is a Messaging connection. Use the JMS connection to read messages from a JMS server. You can create and manage a JMS connection in the Developer tool or through infacmd.

Prerequisites to Create a JMS Connection and a JMS Data Object

Before you create a JMS connection or data object, you must include the JMS provider client libraries on the machine running Data Engineering Streaming.

To create a JMS connections, copy the following JAR files from the IBM MQ server:

- com.ibm.mq.allclient.jar
- com.ibm.mq.axis2.jar
- com.ibm.mq.commonservices.jar
- com.ibm.mq.defaultconfig.jar
- com.ibm.mq.headers.jar
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.jms.Nojndi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.pcf.jar
- fscontext.jar
- jms.jar
- providerutil.jar
- com.ibm.mq.postcard.jar
- com.ibm.mq.soap.jar
- com.ibm.mq.tools.ras.jar
- com.ibm.mq.traceControl.jar
- com.ibm.mqjms.jar
- jndi.jar
- jta.jar
- ldap.jar
- rmm.jar

Place the client JAR files in the following location:

- **Developer tool installation directory:** `<Developer Tool Installation Directory>/clients/ DeveloperClient/connectors/thirdparty/infa.jms/common`

Prerequisites to Use a JMS Connection and a JMS Data Object

Before you use a JMS connection and JMS data object in your streaming mapping, perform the following tasks:

1. Verify that the IBM MQ client is installed on all the cluster nodes.

2. Copy the JAR files from the IBM MQ server.

3. Place the JAR files in the following directory on the machine where the Data Integration Service runs:

   `INFA_HOME/services/shared/hadoop/<Hadoop distribution>/extras/spark-auxjars/`

   For example, `/CDH_5.15/extras/spark-auxjars`

4. Create the `.bindings` file using the IBM MQ Server.

   The `.bindings` file must have the details of the MQ Objects that the data objects in the streaming mappings use.

5. Copy the `.bindings` file to the Developer tool machine.

6. When you create the JMS connection, configure the **Connection URL** property with the location of the `.bindings` file on the Developer tool machine.

   - On Windows, specify the following format for the location: `file:////<Path to .bindings file>`
     For example: `file:////C:/JMS`

   - On Linux, specify the following format for the location: `file:///<Path to .bindings file>`

7. Place the `.bindings` file in the same path on all the cluster nodes.

   The default location of the files is `/etc`

   For example, `file:///etc`

8. Before you run the mapping, update the **Connection URL** property with the location of the `.bindings` file on the cluster nodes.

General Properties

The following table describes the general connection properties for the JMS connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Messaging/JMS.

Connection Properties

The following table describes the connection properties for the JMS connection:

Property	Description
Connection URL	The location and port of the JMS provider on which to connect. For example: `tcp://jndiserverA:61616`
User Name	User name to the connection factory.
Password	The password of the user account that you use to connect to the connection factory.
JNDI Context Factory	The JMS provider specific initial JNDI context factory implementation for connecting to the JNDI service. This value is a fully qualified class name of the Initial Context Factory. For example, the class name of the Initial Context Factory for ActiveMQ is `org.apache.activemq.jndi.ActiveMQInitialContextFactory` For more information, see the documentation of the JMS provider.
JNDI Package Prefixes	A colon-delimited list of package prefixes to use when loading URL context factories. These are the package prefixes for the name of the factory class that will create a URL context factory. For more information about the values, see the documentation of the JMS provider.
JMS Connection Factory	The name of the object in the JNDI server that enables the JMS Client to create JMS connections. For example, `jms/QCF` or `jmsSalesSystem`.

Creating a JMS Connection Using infacmd

You can use the infacmd command line program to create a JMS connection.

To create a JMS connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain name> -pd <domain password> -cn
<connection name> -cid <connection id> -ct Jms -o url='<url_val>' username=<u_val>
password=<pass_val> contextFactory='<cf_val>' packagePrefixes='<pp_val>'
jmsConnectionFactory='<jcf_val>'
```

For more information about the CreateConnection command, see the *Informatica Command Reference*.

JDBC Connection Properties

You can use a JDBC connection to access tables in a database. You can create and manage a JDBC connection in the Administrator tool, the Developer tool, or the Analyst tool.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes JDBC connection properties:

Property	Description
Database Type	The database type.
Name	Name of the connection. The name is not case sensitive and must be unique within the domain. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. The description cannot exceed 765 characters.
User Name	The database user name.
Password	The password for the database user name.

JDBC Driver Class Name	Name of the JDBC driver class.
	The following list provides the driver class name that you can enter for the applicable database type:
	- DataDirect JDBC driver class name for Oracle:
	  `com.informatica.jdbc.oracle.OracleDriver`
	- DataDirect JDBC driver class name for IBM DB2:
	  `com.informatica.jdbc.db2.DB2Driver`
	- DataDirect JDBC driver class name for Microsoft SQL Server:
	  `com.informatica.jdbc.sqlserver.SQLServerDriver`
	- DataDirect JDBC driver class name for Sybase ASE:
	  `com.informatica.jdbc.sybase.SybaseDriver`
	- DataDirect JDBC driver class name for Informix:
	  `com.informatica.jdbc.informix.InformixDriver`
	- DataDirect JDBC driver class name for MySQL:
	  `com.informatica.jdbc.mysql.MySQLDriver`
	- JDBC driver for Databricks Delta Lake: the name of the driver that you downloaded from Databricks. For information about the driver, see the topic on configuring storage access in the "Before You Begin Databricks Integration" chapter of the *Data Engineering Integration Guide*.
	For more information about which driver class to use with specific databases, see the vendor documentation.

Connection String	Connection string to connect to the database. Use the following connection string: `jdbc:<subprotocol>:<subname>` For more information about the connection string to use with specific drivers, see the vendor documentation.
Environment SQL	Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the connection environment SQL each time it connects to the database. **Note:** If you enable Sqoop, Sqoop ignores this property.
Transaction SQL	Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the transaction environment SQL at the beginning of each transaction. **Note:** If you enable Sqoop, Sqoop ignores this property.

Property	Description
SQL Identifier Character	Type of character that the database uses to enclose delimited identifiers in SQL queries. The available characters depend on the database type. Select (None) if the database uses regular identifiers. When the Data Integration Service generates SQL queries, the service does not place delimited characters around any identifiers. Select a character if the database uses delimited identifiers. When the Data Integration Service generates SQL queries, the service encloses delimited identifiers within this character. **Note:** If you enable Sqoop, Sqoop ignores this property.
Support Mixed-case Identifiers	Enable if the database uses case-sensitive identifiers. When enabled, the Data Integration Service encloses all identifiers within the character selected for the **SQL Identifier Character** property. When the **SQL Identifier Character** property is set to none, the **Support Mixed-case Identifiers** property is disabled. **Note:** If you enable Sqoop, Sqoop honors this property when you generate and execute a DDL script to create or replace a target at run time. In all other scenarios, Sqoop ignores this property.
Use Sqoop Connector	Enables Sqoop connectivity for the data object that uses the JDBC connection. The Data Integration Service runs the mapping in the Hadoop run-time environment through Sqoop. You can configure Sqoop connectivity for relational data objects, customized data objects, and logical data objects that are based on a JDBC-compliant database. Select **Sqoop v1.x** to enable Sqoop connectivity. Default is **None**.

Sqoop Arguments	Enter the arguments that Sqoop must use to connect to the database. Separate multiple arguments with a space.
	
	To run the mapping on the Blaze engine with the Teradata Connector for Hadoop (TDCH) specialized connectors for Sqoop, you must define the TDCH connection factory class in the Sqoop arguments. The connection factory class varies based on the TDCH Sqoop Connector that you want to use.
	- To use Cloudera Connector Powered by Teradata, configure the following Sqoop argument:
	
	    `-Dsqoop.connection.factories=com.cloudera.connector.teradata.TeradataManagerFactory`
	- To use Hortonworks Connector for Teradata (powered by the Teradata Connector for Hadoop), configure the following Sqoop argument:
	
	    `-Dsqoop.connection.factories=org.apache.sqoop.teradata.TeradataManagerFactory`
	
	To run the mapping on the Spark engine, you do not need to define the TDCH connection factory class in the Sqoop arguments. The Data Integration Service invokes the Cloudera Connector Powered by Teradata and Hortonworks Connector for Teradata (powered by the Teradata Connector for Hadoop) by default.
	
	Note: To run the mapping with a generic JDBC connector instead of the specialized Cloudera or Hortonworks connector, you must define the --driver and --connection-manager Sqoop arguments in the JDBC connection. If you define the --driver and --connection-manager arguments in the Read or Write transformation of the mapping, Sqoop ignores the arguments.
	
	If you do not enter Sqoop arguments, the Data Integration Service constructs the Sqoop command based on the JDBC connection properties.

JDBC V2 Connection Properties

When you set up a JDBC V2 connection, you must configure the connection properties.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes the JDBC V2 connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters:~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select JDBC V2.

The **Details** tab contains the connection attributes of the JDBC V2 connection. The following table describes the connection attributes:

Property	Description
Username	The database user name. User name with permissions to access the database that supports the Type 4 JDBC driver.
Password	The password for the database user name.
Schema Name	Optional. The schema name to connect in the database. If you do not specify the schema name, all the schemas available in the database are listed.
JDBC Driver Class Name	Name of the JDBC driver class. The following list provides the driver class name that you can enter for the applicable database type: - JDBC driver class name for Azure SQL Database: `com.microsoft.sqlserver.jdbc.SQLServerDriver` - JDBC driver class name for Aurora PostgreSQL: `org.postgresql.Driver` - JDBC driver class name for SAP HANA Database: `com.sap.db.jdbc.Driver` For more information about which driver class to use with specific databases, see the third-party vendor documentation.

Property	Description
Connection String	Connection string to connect to the database. Use the following connection string: `jdbc:<subprotocol>:<subname>` The following list provides sample connection strings that you can enter for the applicable database type: - Connection string for Azure SQL Database JDBC driver: `jdbc:sqlserver://<host>:<port>;database=<database_name>` - Connection string for Aurora PostgreSQL JDBC driver: `jdbc:postgresql://<host>:<port>[/<database_name>]` - Connection string for SAP HANA Database driver: `jdbc:sap://<host>:<port>/?databaseName=<Database_Name>` For more information about the connection string to use with specific drivers, see the third-party vendor documentation.
Sub Type	The database type to which you want to connect. You can select from the following database types to connect: - **Azure SQL Database**. Connects to Azure SQL database. - **PostgreSQL**. Connects to Aurora PostgreSQL database. - **SAP HANA Database**. Connects to SAP HANA database. - **Others** . Connects to any database that supports the Type 4 JDBC driver.
Support Mixed-case Identifiers	Enable if the database uses case-sensitive identifiers. When enabled, the Data Integration Service encloses all identifiers within the character selected for the **SQL Identifier Character** property. For example, Aurora PostgreSQL database supports mixed-cased characters. You must enable this property to connect to the Aurora PostgreSQL database. When the **SQL Identifier Character** property is set to none, the **Support Mixed-case Identifiers** property is disabled.
SQL Identifier Character	Type of character that the database uses to enclose delimited identifiers in SQL queries. The available characters depend on the database type. Select (None) if the database uses regular identifiers. When the Data Integration Service generates SQL queries, the service does not place delimited characters around any identifiers. Select a character if the database uses delimited identifiers. When the Data Integration Service generates SQL queries, the service encloses delimited identifiers within this character. **Note:** Select **SQL Identifier Character** as **None** when you specify the SAP HANA Database subtype.

Kafka Connection Properties

The Kafka connection is a Messaging connection. Use the Kafka connection to access an Apache Kafka broker as a source or a target. You can create and manage a Kafka connection in the Developer tool or through infacmd.

When you configure a Kafka connection, you configure the following properties:

- The list of Kafka brokers that the connection reads from or writes to.

- The number of seconds the Integration Service attempts to reconnect to the database if the connection fails.

- Version of the Kafka messaging broker. Configure the Kafka messaging broker version to Apache 0.10.1.1 & above.

General Properties

The following table describes the general connection properties for the Kafka connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) − + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Messaging/Kafka.

Kafka Broker Properties

The following table describes the Kafka broker properties for the Kafka connection:

Property	Description
Kafka Broker List	Comma-separated list of Kafka brokers which maintain the configuration of the Kafka messaging broker. To specify a Kafka broker, use the following format: `<IP Address>:<port>`
Retry Timeout	Number of seconds after which the Integration Service attempts to reconnect to the Kafka broker to read or write data. If the source or target is not available for the time you specify, the mapping execution stops to avoid any data loss.

Property	Description
Kafka Broker Version	Configure the Kafka messaging broker version to Apache 0.10.1.1 & above.
Additional Connection Properties	Optional. Comma-separated list of connection properties to connect to the Kafka broker. For example, you can use the following syntax: `request.timeout.ms=<value>,session.timeout.ms=<value>, fetch.max.wait.ms=<value>,heartbeat.interval.ms=<value >, security.protocol=SASL_PLAINTEXT,sasl.kerberos. service.name=<kerberos name>,sasl.mechanism=GSSAPI, sasl.jaas.config=com.sun.security.auth.module. Krb5Login Modulerequired useKeyTab=true doNotPrompt=true storeKey=true client=true keyTab="<Keytab Location>" principal="<principal>";`  To reduce the time taken to connect to the Kafka broker, ensure that you set the following properties: - `request.timeout.ms` - `session.timeout.ms` - `fetch.max.wait.ms` - `heartbeat.interval.ms`  To connect to the Kafka broker in a secured way, ensure that you set one of the following values for the `security.protocol` property: - SASL_SSL - SSL  The default value of `security.protocol` property is SASL_PLAINTEXT. **Technical Preview:** The Additional Connection Properties is available for technical preview. Technical preview functionality is supported but is unwarranted and is not production-ready. Informatica recommends that you use in non-production environments only. For more information about the connection properties, see https://kafka.apache.org/documentation/.

SSL Properties

The following table describes the SSL properties for the Kafka connection:

Property	Description
SSL Mode	Required. SSL mode indicates the encryption type to use for the connection. You can choose a mode from the following SSL modes: - Disabled - One way - Two way
SSL TrustStore File Path	Required when **One way** SSL mode is selected. Absolute path and file name of the SSL truststore file that contains certificates of the trusted SSL server.
SSL TrustStore Password	Required when **One way** SSL mode is selected. Password for the SSL truststore.

Property	Description
SSL KeyStore File Path	Required when **Two way** SSL mode is selected. Absolute path and file name of the SSL keystore file that contains private keys and certificates for the SSL server.
SSL KeyStore Password	Required when **Two way** SSL mode is selected. Password for the SSL keystore.

Creating a Kafka Connection Using infacmd

You can use the infacmd command line program to create a Kafka connection.

To create a Kafka connection on UNIX, run the following command:

```
sh infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -cid <connection id> -ct Kafka -o
kfkBrkList=<host1:port1>,<host2:port2>,<host3:port3> kafkabrokerversion=<version>
additionalConnectionProperties=<additional properties>
```

For more information about the CreateConnection command, see the *Informatica Command Reference*.

Kudu Connection Properties

Use a Kudu connection to access Kudu.

Note: The order of the connection properties might vary depending on the tool where you view them.

You can create and manage a Kudu connection in the Administrator tool or the Developer tool. The following table describes the Kudu connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select Kudu.

The following table describes the properties for metadata access:

Property	Description
Kudu Master URLs	The URLs of the Kudu master tables.
Kudu Library Version	The version number of the Kudu library.
Cluster Configuration	The Hadoop cluster that you use for the connection.

Creating a Kudu Connection using Informatica Command Line Program

You can use the infacmd command line program to create a Kudu connection.

To create a Kudu connection on UNIX, run the following command:

```
./infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <domain password> -cn
<connection name> -ct Kudu -o kuduMasterUrls=<host1:port1>,<host2:port2>
kuduTplVersion="1.11.0" clusterConfigId=<Cluster ConfigId>
```

For more information about the CreateConnection command and Kudu connection options, see the *Informatica Command Reference*.

MapR Streams Connection

The MapR Streams connection is a Messaging connection. Use the MapR Streams connection to access MapR Streams as source or target. You can create and manage a MapR Streams connection in the Developer tool or through infacmd.

When you configure an MapR Streams connection, you configure the path to the MapR Stream and the name of the MapR Stream.

Before you create and use MapR Streams connections in Streaming mappings, complete the required prerequisites. For more information about the prerequisite tasks, see the *Data Engineering Integration Guide*.

General Properties

The following table describes the general connection properties for the MapR Streams connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.

Property	Description
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Messaging/MapR Streams.

Connection Properties

The following table describes the connection properties for the MapR Streams connection:

Property	Description
Stream Path and Name	The path that contains the MapR Stream and the name of the MapR Stream. Use the following format: `/stream-path:stream-name`
Connection Timeout (in msec)	Number of milliseconds that the Integration service waits to establish a connection to MapR Stream after which it times out.

Creating a MapR Streams Connection Using infacmd

You can use the infacmd command line program to create a MapR Streams connection.

To create a MapR Streams connection on UNIX, run the following command:

```
infacmd createConnection -dn <domain name> -un <domain user> -pd<password> -cn <connection
name> -cid <connection id> -ct MAPRSTREAMS -o connRetryTimeout=10000 streamName=/sample-
stream
```

For more information about the CreateConnection command, see the *Informatica Command Reference*.

Microsoft Azure Data Lake Storage Gen1 Connection

Use the Microsoft Azure Data Lake Storage Gen1 connection to access Microsoft Azure Data Lake Storage Gen1 tables as targets. You can create and manage a Microsoft Azure Data Lake Storage Gen1 connection in the Developer tool.

You cannot run a mapping with a Microsoft Azure Data Lake Storage Gen1 as target on a MapR distribution.

Microsoft Azure Data Lake Storage Gen1 Connection Properties

Use a Microsoft Azure Data Lake Storage Gen1 connection to access a Microsoft Azure Data Lake Storage Gen1.

Note: The order of the connection properties might vary depending on the tool where you view them.

You can create and manage a Microsoft Azure Data Lake Storage Gen1 connection in the Administrator tool or the Developer tool. The following table describes the Microsoft Azure Data Lake Storage Gen1 connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}]	\ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select Microsoft Azure Data Lake Storage Gen1.

The following table describes the properties for metadata access:

Property	Description
ADLS Account Name	The name of the Microsoft Azure Data Lake Storage Gen1.
ClientID	The ID of your application to complete the OAuth Authentication in the Active Directory.
Client Secret	The client secret key to complete the OAuth Authentication in the Active Directory.
Directory	The Microsoft Azure Data Lake Storage Gen1 directory that you use to read data or write data. The default is root directory.
AuthEndpoint	The OAuth 2.0 token endpoint from where access code is generated based on based on the Client ID and Client secret is completed.

For more information about creating a client ID, client secret, and auth end point, contact the Azure administrator or see Microsoft Azure Data Lake Storage Gen1 documentation.

Microsoft Azure Data Lake Storage Gen2 Connection

Use the Microsoft Azure Data Lake Storage Gen2 connection to access Microsoft Azure Data Lake Storage Gen2 tables as targets. You can create and manage a Microsoft Azure Data Lake Storage Gen2 connection in the Developer tool.

General Properties

You can create and manage a Microsoft Azure Data Lake Storage Gen2 connection in the Administrator tool or the Developer tool.

The following table describes the Microsoft Azure Data Lake Storage Gen2 general connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}]	\ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select Microsoft Azure Data Lake Storage Gen2.

Connection Properties

The following table describes the Microsoft Azure Data Lake Storage Gen2 connection properties:

Property	Description
Account Name	The name of the Microsoft Azure Data Lake Storage Gen2 account.
Client ID	The ID of your application to complete the OAuth Authentication in the Active Directory.
Client Secret	The client secret key to complete the OAuth Authentication in the Active Directory.
Tenant ID	The tenant ID under which your application resides.
File System Name	The file name and path that you use to write data.
Directory Path	The Microsoft Azure Data Lake Storage Gen2 directory that you use to write data. The default is root directory.

For more information about creating a client ID, client secret, and tenant ID, contact the Azure administrator or see Microsoft Azure Data Lake Storage Gen2 documentation.

Creating a Microsoft Azure Data Lake Storage Gen2 Connection Using infacmd

You can use the infacmd command line program to create a Microsoft Azure Data Lake Storage Gen2 connection.

To create a Microsoft Azure Data Lake Storage Gen2 connection on UNIX, run the following command:

```
infacmd.sh createConnection -dn <domain name> -un <domain user> -pd <password> -cn
<connection name> -ct ADLSGEN2 -o "ACCOUNTNAME=<account name> CLIENTID='<access key ID>'
CLIENTSECRET='<secret access key>' TENANTID='<tenant ID>' FILESYSTEMNAME='<file system name>'
DIRECTORYPATH='<directory path>'"
```

For more information about the CreateConnection command, see the *Informatica Command Reference*.

Snowflake Connection

Create a Snowfake connection to write data to Snowflake.

You can create and manage a Snowflake connection in the Developer tool or through `infacmd`.

General Properties

The following table describes the general connection properties for the Snowfalke connection:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	The string that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. Enter a string that you can use to identify the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection. Select the domain name.
Type	The connection type. Select Snowflake.

Snowflake Connection Properties

When you set up a Snowflake connection, you must configure the connection properties.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes the Snowflake connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters:~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. The ID must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	Optional. The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select SnowFlake.
Username	The user name to connect to the Snowflake account.
Password	The password to connect to the Snowflake account.
Account	The name of the Snowflake account.
Warehouse	The Snowflake warehouse name.
Role	The Snowflake role assigned to the user.

Additional JDBC URL Parameters	Enter one or more JDBC connection parameters in the following format:
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	For example:
	`user=jon&warehouse=mywh&db=mydb&schema=public`
	To access Snowflake through Okta SSO authentication, enter the web-based IdP implementing SAML 2.0 protocol in the following format:
	`authenticator=https://<Your_Okta_Account_Name>.okta.com`
	Note: Microsoft ADFS is not supported.
	For more information about configuring Okta authentication, see the following website: https://docs.snowflake.net/manuals/user-guide/admin-security-fed-auth-configure-snowflake.html#configuring-snowflake-to-use-federated-authentication

Creating a Snowflake Connection using Informatica Command Line Program

You can use the infacmd command line program to create a Snowflake connection.

To create a Snowflake connection on UNIX, run the following command:

```
./infacmd.sh createconnection -dn Domain_Snowflake -un Administartor -pd Administrator -cn
Snowflake_CLI -ct SNOWFLAKE -o "user=INFAADPQA password=passwd account=informatica
role=ROLE_PC_AUTO warehouse=QAAUTO_WH"
```

For more information about the CreateConnection command and Snowflake connection options, see the *Informatica Command Reference*.

Monitoring REST API Reference

This appendix includes the following topic:

Monitoring REST API Reference

You can monitor statistics retrieved through the REST API.

With the REST API, you can get the mapping execution statistics for the streaming mappings that run on the Spark engine. You can get mapping statistics, mapping advanced statistics, mapping execution steps, and mapping execution plans.

For more information, see the *Data Engineering Administrator Guide*.

Sample Files

This appendix includes the following topic:

-

Sample Files

The data objects in a streaming mapping read and write data in XML, JSON, Avro, CSV, and Parquet format. The following examples contain samples for each schema format.

Sample XSD File

When you configure the data operation properties, specify the format in which the data object reads or writes data. When you specify XML format, provide an XSD.

The following sample XSD describes the elements in an XML file:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"/>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Sample JSON Schema

When you configure the data operation properties, specify the format in which the data object reads or writes data. When you specify JSON format, provide a sample JSON file.

The following file is a sample JSON file:

```
{
                "GlossaryCont": {
                                "title": {
                                                "string": "glossaryContTitle0"
                                },
                                "glossDiv": {
                                                "title": "glossaryDivTitle0",
                                                "glossList": {
                                                                "glossEntry": {
                                                                                "id":
"ID0",

"sortAs": "sortAS0",

"glossTerm": "GlossTerm0",

"acronym": "Acronym0",

"abbrev": "Abbrev0",

"glossSee": "GlossSee0",

"glossInteger": 0,

"glossDouble": 0.234,

"glossLong": 100000000000000000,

"glossDef": {

"para": "para0",

"glossSeeAlso": ["glossSeeAlso_0_0"]
                                                                                }
                                                                }
                                                }
                                }
                }
}
```

Sample Avro Schema

When you configure the data operation properties, specify the format in which the data object reads or writes data. When you specify Avro format, provide a sample Avro schema in a .avsc file.

The following file is a sample Avro schema:

```
{
  "type" : "record",
  "name" : "Tree",
  "fields" : [
{"name" : "children", "type" : ("type" : "array", "items": "Tree"}}
]}
```

Sample Flat Format Schema

When you configure the data operation properties, specify the format in which the data object reads or writes data. When you specify Flat format, provide a sample CSV file.

The following file is a sample CSV file:

```
acc_no,acc_type
1,Savings
2,Savings
```

Sample Parquet Schema

When you configure the data operation properties, specify the format in which the data object writes data.

The following file is a sample Parquet schema:

```
message emp_schema {

  optional int32 EmpID;

  optional binary LName (UTF8);
  optional binary FName (UTF8);
   optional double salary;
  optional int32 age;


}
```

Index