Informatica®

Informatica® Data Engineering Integration
10.5.6

Data Engineering Administrator Guide

Informatica Data Engineering Integration Data Engineering Administrator Guide
10.5.6
May 2024

Subject to your opt-out rights, the software will automatically transmit to Informatica in the USA information about the computing and network environment in which the Software is deployed and the data usage and system statistics of the deployment. This transmission is deemed part of the Services under the Informatica privacy policy and Informatica will use and otherwise process this information in accordance with the Informatica privacy policy available at https://www.informatica.com/in/privacy-policy.html. You may disable usage collection in Administrator tool.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2024-06-05

Table of Contents

Chapter 6: Cloud Provisioning Configuration 63

Chapter 7: Data Integration Service Processing. 76

Appendix A: Connections Reference. 87

Preface

Read the *Data Engineering Administrator Guide* to understand data engineering architecture. Learn how to configure and manage security between the domain and the non-native environment. Learn how to tune services and engines for large data set processing, and learn how to configure the domain to connect to a compute cluster.

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.

CHAPTER 1

Introduction to Data Engineering Administration

This chapter includes the following topics:

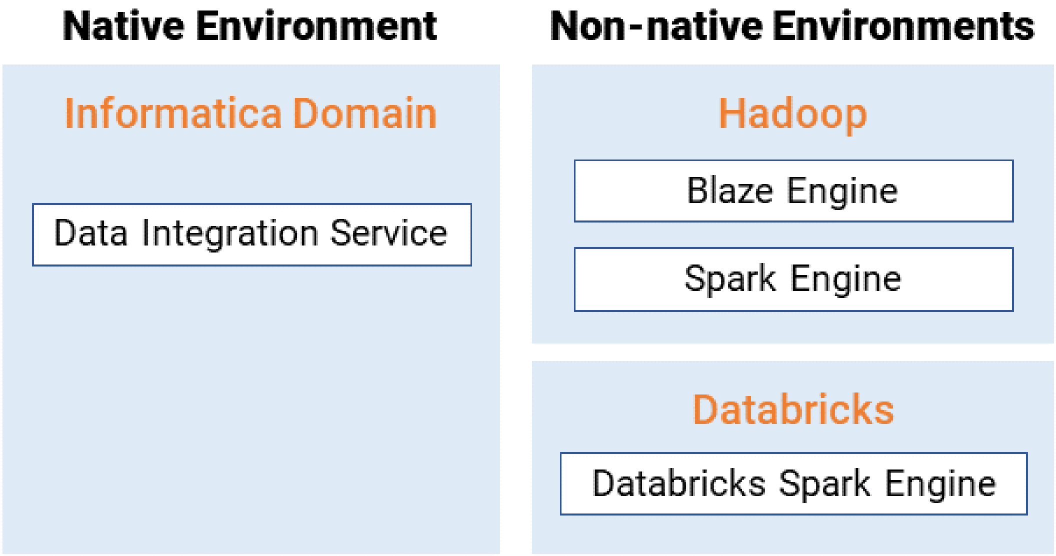
Data Engineering Integration Engines

When you run a mapping, you can choose to run the mapping in the native environment or in a non-native environment, such as Hadoop or Databricks. 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 mapping in the native environment, the Data Integration Service in the Informatica domain runs the mapping. When you run the mapping in a non-native environment, the Data Integration Service pushes the run-time processing to a compute cluster in the non-native environment.

When you run the mapping in a non-native environment, 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.

The following image shows the processing environments and the run-time engines in the environments:



Hadoop Integration

The Informatica domain can connect to clusters that run different Hadoop distributions. Hadoop is an open-source software framework that enables distributed processing of large data sets across clusters of machines. You might also need to use third-party software clients to set up and manage your Hadoop cluster.

The domain can connect to the supported data source in the Hadoop environment, such as HDFS, HBase, or Hive, and push job processing to the Hadoop cluster. To enable high performance access to files across the cluster, you can connect to an HDFS source. You can also connect to a Hive source, which is a data warehouse that connects to HDFS.

It can also connect to NoSQL databases such as HBase, which is a database comprising key-value pairs on Hadoop that performs operations in real-time. The Data Integration Service can push mapping jobs to the Spark or Blaze engine, and it can push profile jobs to the Blaze engine in the Hadoop environment.

Data Engineering Integration supports more than one version of some Hadoop distributions. By default, the cluster configuration wizard populates the latest supported version.

Hadoop Utilities

Data Engineering Integration uses third-party Hadoop utilities such as Sqoop to process data efficiently.

Sqoop is a Hadoop command line program to process data between relational databases and HDFS through MapReduce programs. You can use Sqoop to import and export data. When you use Sqoop, you do not need to install the relational database client and software on any node in the Hadoop cluster.

To use Sqoop, you must configure Sqoop properties in a JDBC connection and run the mapping in the Hadoop environment. You can configure Sqoop connectivity for relational data objects, customized data objects, and logical data objects that are based on a JDBC-compliant database. For example, you can configure Sqoop connectivity for the following databases:

- Aurora
- Greenplum
- IBM DB2
- IBM DB2 for z/OS
- Microsoft SQL Server
- Netezza
- Oracle
- Teradata
- Vertica

The Model Repository Service uses JDBC to import metadata. The Data Integration Service runs the mapping in the Hadoop run-time environment and pushes the job processing to Sqoop. Sqoop then creates map-reduce jobs in the Hadoop cluster, which perform the import and export job in parallel.

Specialized Sqoop Connectors

When you run mappings through Sqoop, you can use the following specialized connectors:

OraOop

You can use OraOop with Sqoop to optimize performance when you read data from or write data to Oracle. OraOop is a specialized Sqoop plug-in for Oracle that uses native protocols to connect to the Oracle database.

You can configure OraOop when you run Sqoop mappings on the Spark engine.

Teradata Connector for Hadoop (TDCH) Specialized Connectors for Sqoop

You can use the following TDCH specialized connectors for Sqoop to read data from or write data to Teradata:

- Cloudera Connector Powered by Teradata
- Hortonworks Connector for Teradata (powered by the Teradata Connector for Hadoop)
- MapR Connector for Teradata

These connectors are specialized Sqoop plug-ins that Cloudera, Hortonworks, and MapR provide for Teradata. They use native protocols to connect to the Teradata database.

Informatica supports Cloudera Connector Powered by Teradata and Hortonworks Connector for Teradata on the Blaze and Spark engines. When you run Sqoop mappings on the Blaze engine, you must configure these connectors. When you run Sqoop mappings on the Spark engine, the Data Integration Service invokes these connectors by default.

Informatica supports MapR Connector for Teradata on the Spark engine. When you run Sqoop mappings on the Spark engine, the Data Integration Service invokes the connector by default.

Note: For information about running native Teradata mappings with Sqoop, see the *Informatica PowerExchange for Teradata Parallel Transporter API User Guide*.

High Availability

High availability refers to the uninterrupted availability of Hadoop cluster components.

You can use high availability for the following services and security systems in the Hadoop environment on Cloudera CDH, Cloudera CDP, Hortonworks HDP, and MapR Hadoop distributions:

- Apache Ranger
- Apache Ranger KMS
- Apache Sentry
- Cloudera Navigator Encrypt
- HBase
- Hive Metastore
- HiveServer2
- Name node
- Resource Manager

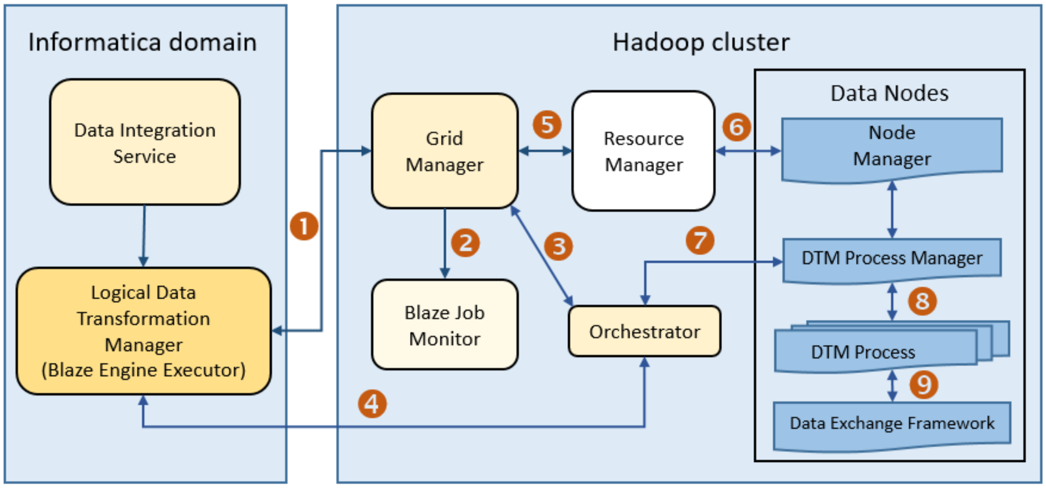
Run-time Process on the Blaze Engine

To run a mapping on the Informatica Blaze engine, the Data Integration Service submits jobs to the Blaze engine executor. The Blaze engine executor is a software component that enables communication between the Data Integration Service and the Blaze engine components on the Hadoop cluster.

The following Blaze engine components appear on the Hadoop cluster:

- Grid Manager. Manages tasks for batch processing.
- Orchestrator. Schedules and processes parallel data processing tasks on a cluster.
- Blaze Job Monitor. Monitors Blaze engine jobs on a cluster.
- DTM Process Manager. Manages the DTM Processes.
- DTM Processes. An operating system process started to run DTM instances.
- Data Exchange Framework. Shuffles data between different processes that process the data on cluster nodes.

The following image shows how a Hadoop cluster processes jobs sent from the Blaze engine executor:



The following events occur when the Data Integration Service submits jobs to the Blaze engine executor:

1. The Blaze Engine Executor communicates with the Grid Manager to initialize Blaze engine components on the Hadoop cluster, and it queries the Grid Manager for an available Orchestrator.

2. The Grid Manager starts the Blaze Job Monitor.
3. The Grid Manager starts the Orchestrator and sends Orchestrator information back to the LDTM.
4. The LDTM communicates with the Orchestrator.
5. The Grid Manager communicates with the Resource Manager for available resources for the Orchestrator.
6. The Resource Manager handles resource allocation on the data nodes through the Node Manager.
7. The Orchestrator sends the tasks to the DTM Processes through the DTM Process Manger.
8. The DTM Process Manager continually communicates with the DTM Processes.
9. The DTM Processes continually communicate with the Data Exchange Framework to send and receive data across processing units that run on the cluster nodes.

Manage Blaze Engines

The Blaze engine remains running after a mapping run. To save resources, you can set a property to stop Blaze engine infrastructure after a specified time period.

Save resources by shutting down Blaze engine infrastructure after a specified time period.

Set the infagrid.blaze.service.idle.timeout property or the infagrid.orchestrator.svc.sunset.time property. You can use the infacmd isp createConnection command, or set the property in the Blaze Advanced properties in the Hadoop connection in the Administrator tool or the Developer tool.

Configure the following Blaze advanced properties in the Hadoop connection:

infagrid.blaze.service.idle.timeout

Optional: The number of minutes that the Blaze engine remains idle before releasing the resources that the Blaze engine uses.

The value must be an integer. Default is 60.

infagrid.orchestrator.svc.sunset.time

Optional: Maximum lifetime for an Orchestrator service, in hours.

You can disable sunset by setting the property to 0 or a negative value. If you disable sunset, the Orchestrator never shuts down during a mapping run.

The value must be an integer. Default is 24 hours.

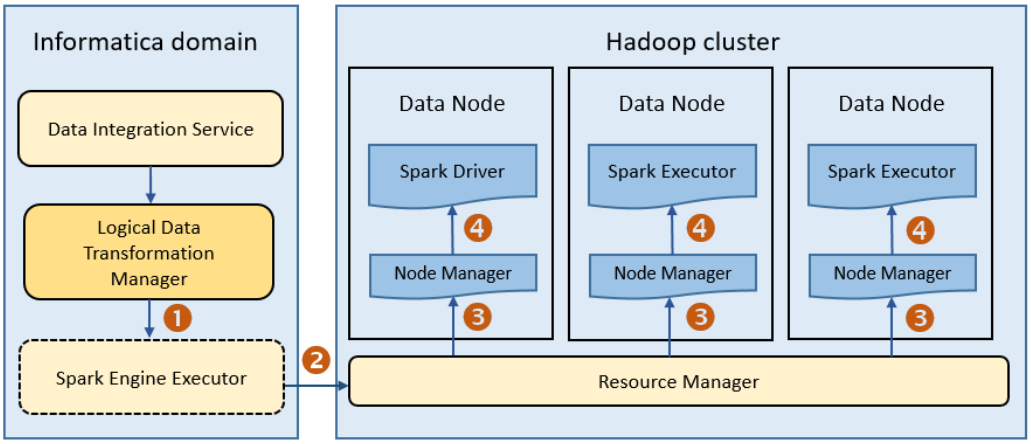
Note: Set the value to less than the Kerberos user ticket maximum renewal time. This Kerberos policy property is usually named like "Maximum lifetime for user ticket renewal."

Run-time Process on the Spark Engine

The Data Integration Service can use the Spark engine on a Hadoop cluster to run Model repository mappings.

To run a mapping on the Spark engine, the Data Integration Service sends a mapping application to the Spark executor. The Spark executor submits the job to the Hadoop cluster to run.

The following image shows how a Hadoop cluster processes jobs sent from the Spark executor:



The following events occur when Data Integration Service runs a mapping on the Spark engine:

1. The Logical Data Transformation Manager translates the mapping into a Scala program, packages it as an application, and sends it to the Spark executor.
2. The Spark executor submits the application to the Resource Manager in the Hadoop cluster and requests resources to run the application.

   Note: When you run mappings on the HDInsight cluster, the Spark executor launches a spark-submit script. The script requests resources to run the application.
3. The Resource Manager identifies the Node Managers that can provide resources, and it assigns jobs to the data nodes.
4. Driver and Executor processes are launched in data nodes where the Spark application runs.

Databricks Integration

The Data Integration Service can push mappings to the Databricks environment. Databricks is an analytics cloud platform that you can use with Microsoft Azure cloud services or Amazon Web Services. Databricks incorporates the open-source Apache Spark cluster technologies and capabilities.

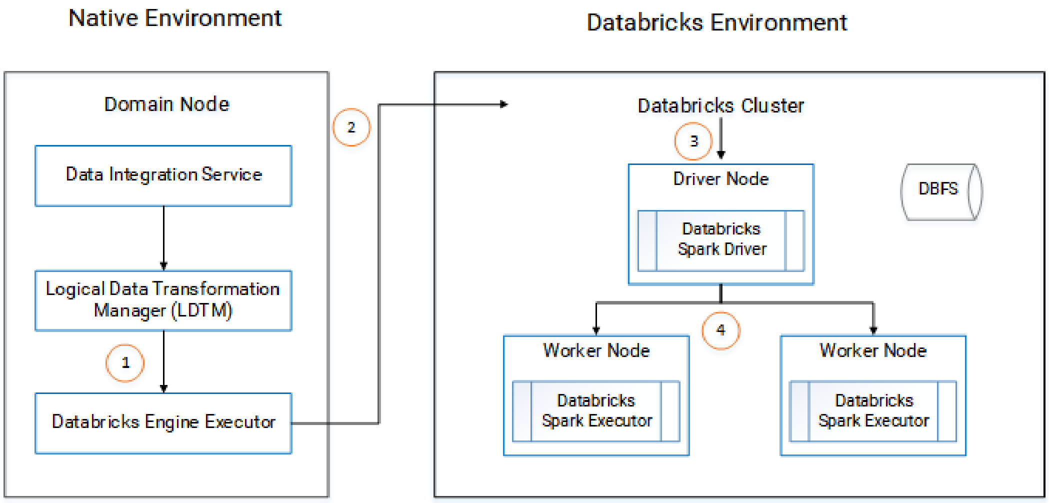
Informatica automatically installs the binaries required to integrate the Informatica domain with the Databricks environment.

The integration requires Informatica connection objects and cluster configurations. A cluster configuration is a domain object that contains configuration parameters that you import from the Databricks cluster. You then associate the cluster configuration with connections to access the Databricks environment.

Run-time Process on the Databricks Spark Engine

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.

The following image shows the components of the Informatica and the Databricks environments:



1. The Logical Data Transformation Manager translates the mapping into a Scala program, packages it as an application, and sends it to the Databricks Engine Executor on the Data Integration Service machine.
2. The Databricks Engine Executor submits the application through REST API to the Databricks cluster, requests to run the application, and stages files for access during run time.
3. The Databricks cluster passes the request to the Databricks Spark driver on the driver node.
4. The Databricks Spark driver distributes the job to one or more Databricks Spark executors that reside on worker nodes.
5. The executors run the job and stage run-time data to the Databricks File System (DBFS) of the workspace.

Application Services

Informatica uses application services in the Informatica domain to process data.

Use the Administrator tool to create connections, monitor jobs, and manage application services.

Data Engineering Integration uses the following application services:

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.

Data Integration Service

The Data Integration Service can process mappings in the native environment or push the mapping for processing to a compute cluster in a non-native environment. The Data Integration Service also retrieves metadata from the Model repository when you run a Developer tool mapping or workflow. The Analyst tool and Developer tool connect to the Data Integration Service to run profile jobs and store profile results in the profiling warehouse.

Mass Ingestion Service

The Mass Ingestion Service manages and validates mass ingestion specifications that you create in the Mass Ingestion tool. The Mass Ingestion Service deploys specifications to the Data Integration Service. When a specification runs, the Mass Ingestion Service generates ingestion statistics.

Metadata Access Service

The Metadata Access Service allows the Developer tool to import and preview metadata from a Hadoop cluster.

The Metadata Access Service contains information about the Service Principal Name (SPN) and keytab information if the Hadoop cluster uses Kerberos authentication. You can create one or more Metadata Access Services on a node. Based on your license, the Metadata Access Service can be highly available.

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, 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, mapping specification, profile, or workflow.

REST Operations Hub

The REST Operations Hub Service is an application service in the Informatica domain that exposes Informatica product functionality to external clients through REST APIs.

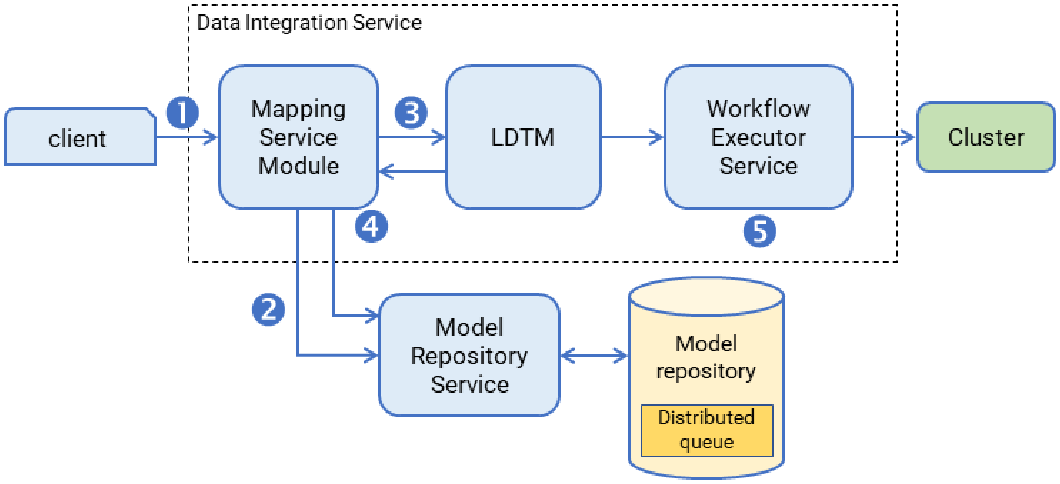
Data Integration Service Process

The Data Integration Service starts one or more Data Integration Service processes to manage requests to run mapping jobs in the Hadoop environment.

When you run mappings in the Hadoop environment, the following components run within the Data Integration Service process:

- Mapping Service Module. The Mapping Service Module receives requests to run mappings from clients.
- Logical Data Transformation Manager (LDTM). The LDTM compiles and optimizes mapping jobs, and it generates the execution workflow that is used to run a mapping on a Hadoop cluster.
- Workflow Executor Service. The Workflow Executor Service is a part of the Data Transformation Manager (DTM). The Data Integration Service uses the Workflow Executor Service to push jobs to a Hadoop cluster.

The following diagram shows how the components interact with the client, the Hadoop cluster, and the Model Repository Service:



1. A client submits a mapping execution request to the Data Integration Service. The Mapping Service Module receives the request and stores the job in the queue.
2. The Mapping Service Module connects to the Model Repository Service to fetch mapping metadata from the Model repository.
3. The Mapping Service Module passes the mapping to the Logical Data Transformation Manager (LDTM).
4. The LDTM compiles the mapping and generates the Spark execution workflow. It stores the execution workflow in the Model repository.
5. The LTDM pushes the execution workflow through the Workflow Executor Service to the cluster for processing.

For more information about the architecture of a Data Integration Service, see the "Data Integration Service Architecture" chapter in the *Informatica Application Service Guide*.

Security

Security for implementations of Data Engineering Integration includes security for the Informatica domain native environment and for the non-native environments.

Security for the Hadoop Environment

You can configure security for the Informatica domain and the Hadoop cluster to protect from threats inside and outside the network. Security for the Hadoop cluster includes the following areas:

Authentication

When the Informatica implementation includes Data Engineering Integration, user identities must be authenticated in the Informatica domain and the Hadoop cluster. Authentication for the Informatica domain is separate from authentication for the Hadoop cluster.

By default, Hadoop does not verify the identity of users. To authenticate user identities, you can configure the following authentication protocols on the cluster:

- Native authentication
- Lightweight Directory Access Protocol (LDAP)

- Kerberos, when the Hadoop distribution supports it
- Apache Knox Gateway

Data Engineering Integration also supports Hadoop clusters that use a Microsoft Active Directory (AD) Key Distribution Center (KDC) or an MIT KDC.

Authorization

After a user is authenticated, a user must be authorized to perform actions. For example, a user must have the correct permissions to access the directories where specific data is stored to use that data in a mapping.

You can run mappings on a cluster that uses one of the following security management systems for authorization:

- Cloudera Navigator Encrypt
- HDFS permissions
- User impersonation
- Apache Ranger
- Apache Sentry
- HDFS Transparent Encryption

Data and metadata management

Data and metadata management involves managing data to track and audit data access, update metadata, and perform data lineage. Data Engineering Integration supports Cloudera Navigator and Metadata Manager to manage metadata and perform data lineage.

Data security

Data security involves protecting sensitive data from unauthorized access. Data Engineering Integration supports data masking with the Data Masking transformation in the Developer tool, Dynamic Data Masking, and Persistent Data Masking.

Operating system profiles

An operating system profile is a type of security that the Data Integration Service uses to run mappings. Use operating system profiles to increase security and to isolate the run-time environment for users. Data Engineering Integration supports operating system profiles on all Hadoop distributions. In the Hadoop run-time environment, the Data Integration Service pushes the processing to the Hadoop cluster and the run-time engines run mappings with the operating system profile.

Security for the Databricks Environment

The Data Integration Service uses token-based authentication to provide access to the Databricks environment. Generate tokens within the Databricks environment and use the token ID to connect to Databricks.

Connections

Informatica requires connections to the non-native environment.

Create and managed the following types of connections within the Administrator tool:

Cluster connection

The connection to the non-native cluster environment. You can create a Hadoop or Databricks cluster connection. The cluster connection must contain a reference either to a cloud provisioning connection or to a cluster configuration.

Cloud provisioning connection

A type of connection that enables the Data Integration Service to contact and create resources on the cloud platform. Create a cloud provisioning configuration when you configure a cluster workflow. You create a cluster workflow to automate the creation of clusters and workflow tasks on an Amazon Web Services, Microsoft Azure, or Azure Databricks cloud platform.

Cluster configuration

An object in the domain that contains configuration information about the compute cluster. The cluster configuration enables the Data Integration Service to push mapping logic to the non-native environment. Import configuration properties from the compute cluster to create a cluster configuration. You can import directly from the cluster or from an archive file. When you create the cluster configuration, you can also choose to create related connections.

CHAPTER 2

Authentication

This chapter includes the following topics:

Authentication Overview

The authentication process verifies the identity of a user account. With Data Engineering Integration, user identities must be authenticated in the native Informatica domain and the non-native environment containing a Hadoop or Databricks cluster. Authentication for the Informatica domain is separate from authentication for the Hadoop cluster.

The Informatica domain uses native, LDAP, and Kerberos authentication. Native authentication stores user credentials and privileges in the domain configuration repository and performs all user authentication within the Informatica domain. LDAP authentication uses an LDAP directory service that stores user accounts and credentials that are accessed over the network.

Hadoop Authentication

By default, Hadoop does not authenticate users. Any user can be used in the Hadoop connection. Informatica recommends that you enable authentication for the cluster. If authentication is enabled for the cluster, the cluster authenticates the user account used for the Hadoop connection between the domain and the cluster.

For a higher level of security, you can set up one of the following types of authentication for the cluster.:

Kerberos authentication

Kerberos is a network authentication protocol that uses tickets to authenticate users and services in a network. Users are stored in the Kerberos principal database, and tickets are issued by a KDC. User impersonation allows different users to run mappings on a Hadoop cluster that uses Kerberos authentication or connect to sources and targets that use Kerberos authentication.

Apache Knox Gateway

The Apache Knox Gateway is a REST API gateway that authenticates users and acts as a single access point for a Hadoop cluster.

For more information about how to enable authentication for the Hadoop cluster, see the documentation for your Hadoop distribution.

Databricks Authentication

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.

Support for Authentication Systems on Hadoop

Depending on the run-time engine that you use, you can run 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 run-time engines supported for the security management system installed on the Hadoop platform:

Hadoop Distribution	Apache Knox	Kerberos	LDAP
Amazon EMR	No support	- Native - Blaze - Spark	- Native - Blaze - Spark
Azure HDInsight	No support	- Native - Blaze - Spark	No support
Cloudera CDH	No support	- Native - Blaze - Spark	- Native - Blaze - Spark
Cloudera CDP Private Cloud	No support	- Native - Blaze - Spark	No support
Cloudera CDP Public Cloud	- Native - Blaze - Spark	- Native - Blaze - Spark	No support
Hortonworks HDP	- Native - Blaze - Spark	- Native - Blaze - Spark	- Native - Blaze - Spark
MapR	No support	- Native - Blaze - Spark	No support

Additional Information for Authentication

Consider the following additional security support statements:

- Informatica supports an Azure HDInsight cluster with Enterprise Security Package. The Enterprise Security Package uses Kerberos for authentication and Apache Ranger for authorization.

- Sqoop cannot access Kerberos-enabled databases.

Authentication with Kerberos

The Informatica domain and the Hadoop cluster can use Kerberos authentication to verify user accounts, when the Hadoop cluster supports Kerberos. You can use Kerberos authentication with the domain, with a supported Hadoop cluster, or with both.

Kerberos is a network authentication protocol that uses tickets to authenticate access to services and nodes in a network. Kerberos uses a Key Distribution Center (KDC) to validate the identities of users and services and to grant tickets to authenticated user and service accounts. Users and services are known as principals. The KDC has a database of principals and their associated secret keys that are used as proof of identity. Kerberos can use an LDAP directory service as a principal database.

You can integrate the Informatica domain with a Kerberos-enabled Hadoop cluster whether the domain is Kerberos-enabled or not.

The requirements for Kerberos authentication for the Informatica domain and for the Hadoop cluster:

Kerberos authentication for the Informatica domain

Kerberos authentication for the Informatica domain requires principals stored in a Microsoft Active Directory (AD) LDAP service. If the Informatica domain is Kerberos-enabled, you must use Microsoft AD for the KDC.

Kerberos authentication for the Hadoop cluster

Informatica supports Hadoop clusters that use an AD KDC or an MIT KDC.

When you enable Kerberos for Hadoop, each user and Hadoop service must be authenticated by the KDC. The cluster must authenticate the Data Integration Service user and, optionally, the Blaze user.

For more information about how to configure Kerberos for Hadoop, see the documentation for your Hadoop distribution.

The configuration steps required for the domain to connect to a Hadoop cluster that uses Kerberos authentication depend on whether the domain uses Kerberos.

User Impersonation

User impersonation allows different users to run mappings in a Hadoop cluster that uses Kerberos authentication or connect to sources and targets that use Kerberos authentication.

The Data Integration Service uses its credentials to impersonate the user accounts designated in the Hadoop connection to connect to the Hadoop cluster or to start the Blaze engine.

When the Data Integration Service impersonates a user account to submit a mapping, the mapping can only access Hadoop resources that the impersonated user has permissions on. Without user impersonation, the Data Integration Service uses its credentials to submit a mapping to the Hadoop cluster. Restricted Hadoop resources might be accessible.

When the Data Integration service impersonates a user account to start the Blaze engine, the Blaze engine has the privileges and permissions of the user account used to start it.

Authentication with Apache Knox Gateway

The Apache Knox Gateway is a REST API gateway that authenticates users and acts as a single access point for a Hadoop cluster.

Knox creates a perimeter around a Hadoop cluster. Without Knox, users and applications must connect directly to a resource in the cluster, which requires configuration on the client machines. A direct connection to resources exposes host names and ports to all users and applications and decreases the security of the cluster.

If the cluster uses Knox, applications use REST APIs and JDBC/ODBC over HTTP to connect to Knox. Knox authenticates the user and connects to a resource.

Configuring Apache Knox for Cloudera CDP Public Cloud

To use Apache Knox authentication with Cloudera CDP Public Cloud clusters, you must configure the Knox IDBroker service.

Add the proxy entries for the keytab user to the Knox IDBroker service that runs on the Data Lake cluster.

For example, add the following entries to the configuration page for *idbroker_kerberos_dt_proxyuser_block*:

```
“hadoop.proxyuser.csso_<keytab user>.groups": "*"
"hadoop.proxyuser.csso_<keytab user>.hosts": "*"
"hadoop.proxyuser.csso_<keytab user>.users": "spn_user"
```

CHAPTER 3

Running Mappings on a Cluster with Kerberos Authentication

This chapter includes the following topics:

Running Mappings with Kerberos Authentication Overview

You can run mappings on a Hadoop cluster that uses MIT or Microsoft Active Directory (AD) Kerberos authentication. Kerberos is a network authentication protocol that uses tickets to authenticate access to services and nodes in a network.

If the Informatica domain uses Kerberos authentication, you must configure a one-way cross-realm trust to enable the Hadoop cluster to communicate with the Informatica domain. The Informatica domain uses Kerberos authentication on an AD service. The Hadoop cluster uses Kerberos authentication on an MIT service. Enable the cross-realm trust to enable the MIT service to communicate with the AD service.

Based on whether the Informatica domain uses Kerberos authentication or not, you might need to perform the following tasks to run mappings on a Hadoop cluster that uses Kerberos authentication:

- If you run mappings in a Hadoop environment, you must configure user impersonation to enable other users to run mappings on the Hadoop cluster.
- If you run mappings in the native environment, you must configure the mappings to read and process data from Hive sources that use Kerberos authentication.
- If you run a mapping that has Hive sources or targets, you must enable user authentication for the mapping on the Hadoop cluster.
- If you import metadata from Hive, complex file sources, and HBase sources, you must configure the Developer tool to use Kerberos credentials to access the Hive, complex file, and HBase metadata.

Running Mappings in a Kerberos-Enabled Hadoop Environment

To run mappings in a Kerberos-enabled Hadoop environment, you must configure the Kerberos configuration file, create user authentication artifacts, and configure Kerberos authentication properties for the Informatica domain. To view metadata from Hive, HBase and complex file sources in the Developer tool, you must import configuration files and generate a Kerberos credentials file.

The Kerberos configuration file `krb5.conf` contains configuration properties for the Kerberos realm. The one-way cross-realm trust enables the Informatica domain to communicate with the Hadoop cluster.

The Informatica domain uses Kerberos authentication on a Microsoft Active Directory service. The Hadoop cluster uses Kerberos authentication on an MIT Kerberos service. You set up a one-way cross-realm trust to enable the KDC for the MIT Kerberos service to communicate with the KDC for the Active Directory service. After you set up the cross-realm trust, you must configure the Informatica domain to enable mappings to run in the Hadoop cluster.

To run mappings on a cluster that uses Kerberos authentication, perform the following configuration tasks:

1. Set up the Kerberos configuration file.
2. When the Informatica domain uses Kerberos authentication, set up the one-way cross-realm trust.
3. Create matching operating system profile user names on each Hadoop cluster node.
4. Create the Service Principal Name and Keytab File in the Active Directory Server.
5. Specify the Kerberos authentication properties for the Data Integration Service.
6. Configure Execution Options for the Data Integration Service.
7. Configure the Developer tool to enable you to view metadata from Hive, HBase and complex file sources when a cluster is Kerberos-enabled.

Step 1. Set Up the Kerberos Configuration File on the Domain Host

Set the properties required by Informatica in the Kerberos configuration file, and then copy the file to each node in the Informatica domain.

`krb5.conf` is located in the `<Informatica Installation Directory>/java/jre/lib/security` directory.

1. Back up `krb5.conf` before you make any changes.
2. Open `krb5.conf` for editing.
3. Configure the following Kerberos library properties in the *libdefaults* section of the file.

The following table describes the properties to enter:

Property	Description
default_realm	Name of the Kerberos realm to which the Informatica domain services belong. The realm name must be in uppercase. If the domain uses a single Kerberos realm for authentication, the service realm name and the user realm name must be the same.
forwardable	Allows a service to delegate client user credentials to another service. The Informatica domain requires application services to authenticate the client user credentials with other services. Set to true.
default_tkt_enctypes	Encryption types for the session key included in ticket-granting tickets (TGT). Set this property only if session keys must use specific encryption types. Ensure that the Kerberos Key Distribution Center (KDC) supports the encryption type that you specify. Do not set this property to allow the Kerberos protocol to select the encryption type to use. If the node hosts or Informatica client hosts use 256-bit encryption, install the Java Cryptography Extension (JCE) unlimited strength policy files on all node hosts and Informatica client hosts to avoid authentication issues.
rdns	Determines whether reverse name lookup is used in addition to forward name lookup to canonicalize host names for use in service principal names. Set to false.
renew_lifetime	The default renewable lifetime for initial ticket requests.
ticket_lifetime	The default lifetime for initial ticket requests.
udp_preference_limit	Determines the protocol that Kerberos uses when it sends a message to the KDC. Set to 1 to use the TCP protocol if the domain experiences intermittent Kerberos authentication failures.
dns_lookup_kdc	Indicates whether the Kerberos client uses DNS SRV records to locate the KDCs and other servers for a realm, if they are not listed in the information for the realm. DNS uses SRV records to identify computers that host specific services. Required when the domain is Kerberos-enabled. Requires you to set the admin_server realm property. Set to true.
dns_lookup_realm	Indicates whether the Kerberos client uses DNS TXT records to determine the Kerberos realm of a host. DNS uses text or TXT records to associate arbitrary text with a host or other name, such as human readable information about a server, network, data center, or other accounting information. Required when the domain is Kerberos-enabled. Set to true.

4. In the *realms* section, set or add the properties required by Informatica.

The following table lists the values to which you must set properties in the realms section:

Property	Description
admin_server	The name or IP address of the Kerberos administration server host. You can include an optional port number, separated from the host name by a colon. Default is 749.
kdc	The name or IP address of a host running the Key Distribution Center (KDC) for the realm. You can include an optional port number, separated from the host name by a colon. Default is 88.

When you use a Kerberos-enabled Cloudera CDP Public Cloud cluster, set both admin_server and kdc to the KDC server IP address. To find the KDC server IP address, run the following command on any cluster node: `ping kdc.<default realm name>`

The following example shows the parameters for the Hadoop realm if the Informatica domain does not use Kerberos authentication:

```
[realms]
HADOOP-REALM = {
     kdc = 123abcdl34.hadoop-realm.com
     admin server = def456.hadoop-realm.com
                                     }
```

The following example shows the parameters for the Hadoop realm if the Informatica domain uses Kerberos authentication:

```
[realms]
INFA-AD-REALM = {
     kdc = 123abcd.infa-realm.com
     admin server = 123abcd.infa-realm.com
                                     }
HADOOP-REALM = {
     kdc = 123abcdl34.hadoop-realm.com
     admin server = def456.hadoop-realm.com
                                     }
```

5. In the *domain_realms* section, map the domain name or host name to a Kerberos realm name. The domain name is prefixed by a period (.).

   The following example shows the parameters for the Hadoop domain_realm if the Informatica domain does not use Kerberos authentication:

   ```
   [domain_realm]
    .hadoop_realm.com = HADOOP-REALM
     hadoop_realm.com = HADOOP-REALM
   ```

   The following example shows the parameters for the Hadoop domain_realm if the Informatica domain uses Kerberos authentication:

   ```
   [domain_realm]
   .infa_ad_realm.com = INFA-AD-REALM
     infa_ad_realm.com = INFA-AD-REALM
   .hadoop_realm.com = HADOOP-REALM
     hadoop_realm.com = HADOOP-REALM
   ```

6. Copy the `krb5.conf` file to the following locations on the machine that hosts the Data Integration Service:

   - `<Informatica installation directory>/services/shared/security/`
   - `<Informatica installation directory>/java/jre/lib/security`

The following example shows the content of a Kerberos configuration file with the required properties for a single Kerberos realm configuration:

```
[libdefaults]
default_realm = COMPANY.COM
forwardable = true
rdns = false
renew_lifetime = 7d
ticket_lifetime = 24h
udp_preference_limit = 1
dns_lookup_kdc = true
dns_lookup_realm = true

[realms]
COMPANY.COM = {
admin_server = KDC01.COMPANY.COM:749
kdc = KDC01.COMPANY.COM:88
}

[domain_realm]
.company.com = COMPANY.COM
company.com = COMPANY.COM
```

The following example shows the content of a Kerberos configuration file with the required properties for a Kerberos cross realm configuration:

```
[libdefaults]
default_realm = COMPANY.COM
forwardable = true
rdns = false
renew_lifetime = 7d
ticket_lifetime = 24h
udp_preference_limit = 1
dns_lookup_kdc = true
dns_lookup_realm = true

[realms]
COMPANY.COM = {
admin_server = KDC01.COMPANY.COM:749
kdc = KDC01.COMPANY.COM:88
}
EAST.COMPANY.COM = {
kdc = 10.75.141.193
admin_server = 10.75.141.193
}
WEST.COMPANY.COM = {
kdc = 10.78.140.111
admin_server = 10.78.140.111

[domain_realm]
.company.com = COMPANY.COM
company.com = COMPANY.COM
.east.company.com = EAST.COMPANY.COM
east.company.com = EAST.COMPANY.COM
.west.company.com = WEST.COMPANY.COM
west.company.com = WEST.COMPANY.COM
```

For more information about the Kerberos configuration file, see the Kerberos network authentication documentation.

Step 2. Set up the Cross-Realm Trust

Perform this step when the Informatica domain uses Kerberos authentication.

Set up a one-way cross-realm trust to enable the KDC for the MIT Kerberos server to communicate with the KDC for the Active Directory server. When you set up the one-way cross-realm trust, the Hadoop cluster can authenticate the Active Directory principals.

To set up the cross-realm trust, you must complete the following steps:

1. Configure the Active Directory server to add the local MIT realm trust.
2. Configure the MIT server to add the cross-realm principal.
3. Translate principal names from the Active Directory realm to the MIT realm.

Configure the Microsoft Active Directory Server

Add the MIT KDC host name and local realm trust to the Active Directory server.

To configure the Active Directory server, complete the following steps:

1. Enter the following command to add the MIT KDC host name:

   ```
   ksetup /addkdc <mit_realm_name> <kdc_hostname>
   ```

   For example, enter the command to add the following values:

   ```
   ksetup /addkdc HADOOP-MIT-REALM def456.hadoop-mit-realm.com
   ```

2. Enter the following command to add the local realm trust to Active Directory:

   ```
   netdom trust <mit_realm_name> /Domain:<ad_realm_name> /add /realm /
   passwordt:<TrustPassword>
   ```

   For example, enter the command to add the following values:

   ```
   netdom trust HADOOP-MIT-REALM /Domain:INFA-AD-REALM /add /realm /passwordt:trust1234
   ```

3. Enter the following commands based on your Microsoft Windows environment to set the proper encryption type:
   For Microsoft Windows 2008, enter the following command:

   ```
   ksetup /SetEncTypeAttr <mit_realm_name> <enc_type>
   ```

   For Microsoft Windows 2003, enter the following command:

   ```
   ktpass /MITRealmName <mit_realm_name> /TrustEncryp <enc_type>
   ```

   Note: The enc_type parameter specifies AES, DES, or RC4 encryption. To find the value for enc_type, see the documentation for your version of Windows Active Directory. The encryption type you specify must be supported on both versions of Windows that use Active Directory and the MIT server.

Configure the MIT Server

Configure the MIT server to add the cross-realm krbtgt principal. The krbtgt principal is the principal name that a Kerberos KDC uses for a Windows domain.

Enter the following command in the kadmin.local or kadmin shell to add the cross-realm krbtgt principal:

```
kadmin:  addprinc -e "<enc_type_list>" krbtgt/<mit_realm_name>@<MY-AD-REALM.COM>
```

The enc_type_list parameter specifies the types of encryption that this cross-realm krbtgt principal will support. The krbtgt principal can support either AES, DES, or RC4 encryption. You can specify multiple encryption types. However, at least one of the encryption types must correspond to the encryption type found in the tickets granted by the KDC in the remote realm.

For example, enter the following value:

```
kadmin:  addprinc -e "rc4-hmac:normal des3-hmac-sha1:normal" krbtgt/HADOOP-MIT-
REALM@INFA-AD-REALM
```

Translate Principal Names from the Active Directory Realm to the MIT Realm

To translate the principal names from the Active Directory realm into local names within the Hadoop cluster, you must configure the hadoop.security.auth_to_local property in the core-site.xml file and hadoop.kms.authentication.kerberos.name.rules property in the kms-site.xml file on all the machines in the Hadoop cluster.

For example, set the following property in core-site.xml on all the machines in the Hadoop cluster:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[1:$1@$0](^.*@INFA-AD-REALM$)s/^(.*)@INFA-AD-REALM$/$1/g
    RULE:[2:$1@$0](^.*@INFA-AD-REALM$)s/^(.*)@INFA-AD-REALM$/$1/g
    DEFAULT
  </value>
</property>
```

For example, set the following property in kms-site.xml on all the machines in the Hadoop cluster:

```
<property>
  <name>hadoop.kms.authentication.kerberos.name.rules</name>
  <value>
    RULE:[1:$1@$0](^.*@INFA-AD-REALM$)s/^(.*)@INFA-AD-REALM$/$1/g
    RULE:[2:$1@$0](^.*@INFA-AD-REALM$)s/^(.*)@INFA-AD-REALM$/$1/g
    DEFAULT
  </value>
</property>
```

Step 3. Create Matching Operating System Profile Names

Create matching operating system profile user names on the machine that runs the Data Integration Service and each Hadoop cluster node to run Informatica mapping jobs.

For example, if user joe runs the Data Integration Service on a machine, you must create the user joe with the same operating system profile on each Hadoop cluster node.

Open a UNIX shell and enter the following UNIX command to create a user with the user name joe.

Step 4. Create the Principal Name and Keytab Files in the Active Directory Server

Create an SPN in the KDC database for Microsoft Active Directory service that matches the user name of the user that runs the Data Integration Service. Create a keytab file for the SPN on the machine on which the KDC server runs. Then, copy the keytab file to the machine on which the Data Integration Service runs.

You do not need to use the Informatica Kerberos SPN Format Generator to generate a list of SPNs and keytab file names. You can create your own SPN and keytab file name.

To create an SPN and Keytab file in the Active Directory server, complete the following steps:

Create a user in the Microsoft Active Directory Service.

Login to the machine on which the Microsoft Active Directory Service runs and create a user with the same name as the user you created in "Step 3. Create Matching Operating System Profile Names" on page 31.

Create an SPN associated with the user.

Use the following guidelines when you create the SPN and keytab files:

- The user principal name (UPN) must be the same as the SPN.
- Enable delegation in Microsoft Active Directory.
- Use the ktpass utility to create an SPN associated with the user and generate the keytab file.

For example, enter the following command:

```
ktpass -out infa_hadoop.keytab -mapuser joe -pass tempBG@2008 -princ joe/
domain12345@INFA-AD-REALM -crypto all
```

Note: The -out parameter specifies the name and path of the keytab file. The -mapuser parameter is the user to which the SPN is associated. The -pass parameter is the password for the SPN in the generated keytab. The -princ parameter is the SPN.

- Use the ktutil utility to generate the keytab file for an Azure HDInsight cluster that uses Enterprise Security Package and ADLS storage.
  For example, enter the following command:

```
sshuser@hn0-hivesc:/tmp/keytabs$ ktutil
ktutil: addent -password -p alice -k 1 -e RC4-HMAC
Password for alice@SECUREHADOOPRC.ONMICROSOFT.COM;
ktutil: wkt /tmp/keytabs/alice.keytab
ktutil: q
```

Step 5. Specify the Kerberos Authentication Properties for the Data Integration Service

In the Data Integration Service properties, configure the properties that enable the Data Integration Service to connect to a Hadoop cluster that uses Kerberos authentication. Use the Administrator tool to set the Data Integration Service properties.

Property	Description
Cluster Staging Directory	The directory on the cluster where the Data Integration Service pushes the binaries to integrate the native and non-native environments and to store temporary files during processing. Default is /tmp.
Hadoop Staging User	The HDFS user that performs operations on the Hadoop staging directory. The user requires write permissions on Hadoop staging directory. Default is the operating system user that starts the Informatica daemon.
Custom Hadoop OS Path	The local path to the Informatica server binaries compatible with the Hadoop operating system. Required when the Hadoop cluster and the Data Integration Service are on different supported operating systems. The Data Integration Service uses the binaries in this directory to integrate the domain with the Hadoop cluster. The Data Integration Service can synchronize the following operating systems: - SUSE and Redhat Include the source directory in the path. For example, <Informatica server binaries>/source. Changes take effect after you recycle the Data Integration Service. **Note:** When you install an Informatica EBF, you must also install it in this directory.
Hadoop Kerberos Service Principal Name	Service Principal Name (SPN) of the Data Integration Service to connect to a Hadoop cluster that uses Kerberos authentication. Not required for the MapR distribution.

Property	Description
Hadoop Kerberos Keytab	The file path to the Kerberos keytab file on the machine on which the Data Integration Service runs. Not required for the MapR distribution.
Custom Properties	Properties that are unique to specific environments. You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities: 1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option 2. Mapping run-time properties for the Hadoop environment 3. Hadoop connection advanced properties for run-time engines 4. Hadoop connection advanced general properties, environment variables, and classpaths 5. Data Integration Service custom properties **Note:** When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez`

Step 6. Configure the Execution Options for the Data Integration Service

To determine whether the Data Integration Service runs jobs in separate operating system processes or in one operating system process, configure the Launch Job Options property. Use the Administrator tool to configure the execution options for the Data Integration Service.

1. Click **Edit** to edit the **Launch Job Options** property in the execution options for the Data Integration Service properties.
2. Choose the launch job option.
   - If you configure the Data Integration Service to launch jobs as a separate process, you must specify the location of the `krb5.conf` file in the Java Virtual Manager (JVM) Options as a custom property in the Data Integration Service process. `krb5.conf` is located in the following directory:`<Informatica Installation Directory>/java/jre/lib/security`.
   - If you configure the Data Integration Service to launch jobs in the service process, you must specify the location of krb5.conf in the **Java Command Line Options** property in the Advanced Properties of the Data Integration Service process. Use the following syntax:

     ```
     -Djava.security.krb5.conf=<Informatica installation directory>/java/jre/lib/
     security/krb5.conf
     ```

Step 7. Configure the Developer Tool

To import metadata from Hive, HBase, and complex file sources, import configuration files from the Kerberos-enabled cluster, and generate the Kerberos credentials file on the Developer tool machine.

Import configuration files

The Hadoop cluster uses a set of XML files named *-site.xml to store configuration settings. The domain uses the same set of files to create the cluster configuration object.

To enable you to import metadata from the cluster, import the *-site.xml files to each Developer tool machine:

1. Log in to the Administrator tool and navigate to **Connections** > **Cluster Configuration** > **CCO**. Locate the cluster configuration associated with the Hadoop cluster.
2. Extract the *-site.xml files in the cluster configuration, inlcuding sensitive properties, to the following directory on the Developer tool machine: `<Informatica installation directory>\clients\DeveloperClient\hadoop\<Hadoop distribution>\conf`
   For more information about sensitive properties, see "Active Properties View" on page 48.

Note: If you refresh the cluster configuration, repeat these steps.

Generate the Kerberos credentials file

1. Copy the krb5.conf file from `<Developer tool installation directory>/services/shared/security` to `C:/Windows`.
2. Rename krb5.conf to krb5.ini.
3. In the krb5.ini file, verify the value of the forwardable option to determine how to use the kinit command. If `forwardable=true`, run the command with the `-f` option. Otherwise, run the command without the -f option.
4. To generate the Kerberos credentials file, run the kinit command from the following location: `<Developer tool installation directory>/clients/java/bin/kinit.exe`
   For example, you might run the following command: `kinit joe/domain12345@MY-REALM`

User Impersonation with Kerberos Authentication

You can enable different users to run mappings in a Hadoop cluster that uses Kerberos authentication or connect to sources and targets that use Kerberos authentication. To enable different users to run mappings or connect to sources and targets, you must configure user impersonation.

You can configure user impersonation for the native or Hadoop environment.

Before you configure user impersonation, you must complete the following prerequisites:

- Complete the tasks for running mappings in a Kerberos-enabled Hadoop environment.
- Configure Kerberos authentication for the native or Hadoop environment.
- If the Hadoop cluster uses MapR, create a proxy directory for the user who will impersonate other users.

If the Hadoop cluster does not use Kerberos authentication, you can specify a user name in the Hadoop connection to enable the Data Integration Service to impersonate that user.

If the Hadoop cluster uses Kerberos authentication, you must specify a user name in the Hadoop connection.

User Impersonation in the Hadoop Environment

To enable different users to run mapping and workflow jobs on a Hadoop cluster that uses Kerberos authentication, you must configure user impersonation in the Hadoop environment.

For example, you want to enable user Bob to run mappings and workflows on the Hadoop cluster that uses Kerberos authentication.

To enable user impersonation, you must complete the following steps:

1. In the Active Directory, enable delegation for the Service Principal Name for the Data Integration Service to enable Bob to run Hadoop jobs.
2. If the service principal name (SPN) is different from the impersonation user, grant read permission on Hive tables to the SPN user.
3. Specify Bob as the user name in the Hadoop connection.

Note: To run a kinit command in a Command task, use a custom Kerberos credential cache file location to separate users. This allows the Data Integration Service to run mappings in parallel and isolate the internal Kerberos authentication.

User Impersonation in the Native Environment

To enable different users to run mappings that read or processes data from sources or targets that use Kerberos authentication, configure user impersonation for the native environment.

To enable user impersonation, you must complete the following steps:

1. Specify Kerberos authentication properties for the Data Integration Service.
2. Configure the execution options for the Data Integration Service.

Step 1. Specify the Kerberos Authentication Properties for the Data Integration Service

In the Data Integration Service properties, configure the properties that enable the Data Integration Service to connect to a Hadoop cluster that uses Kerberos authentication. Use the Administrator tool to set the Data Integration Service properties.

Description	Property
Hadoop Kerberos Service Principal Name	Service Principal Name (SPN) of the Data Integration Service to connect to a Hadoop cluster that uses Kerberos authentication. Not required for the MapR distribution.
Hadoop Kerberos Keytab	The file path to the Kerberos keytab file on the machine on which the Data Integration Service runs. Not required for the MapR distribution.

Step 2. Configure the Execution Options for the Data Integration Service

To determine whether the Data Integration Service runs jobs in separate operating system processes or in one operating system process, configure the **Launch Job Options** property. Use the Administrator tool to configure the execution options for the Data Integration Service.

1. Click **Edit** to edit the **Launch Job Options** property in the execution options for the Data Integration Service properties.
2. Choose the launch job option.
    - If you configure the Data Integration Service to launch jobs as a separate process, you must specify the location of the krb5.conf in the Java Virtual Manager (JVM) Options as a custom property in the Data Integration Service process. krb5.conf is located in the following directory:`<Informatica Installation Directory>/java/jre/lib/security`.

- If you configure the Data Integration Service to launch jobs in the service process, you must specify the location of krb5.conf in the **Java Command Line Options** property in the Advanced Properties of the Data Integration Service process. Use the following syntax:

  ```
  -Djava.security.krb5.conf=<Informatica installation directory>/java/jre/lib/
  security/krb5.conf
  ```

  .

Running Mappings in the Native Environment

To read and process data from Hive, HBase, or HDFS sources that use Kerberos authentication, you must configure Kerberos authentication for mappings in the native environment.

To read and process data from Hive, HBase, or HDFS sources, perform the following steps:

1. Complete the tasks for running mappings in a Kerberos-enabled Hadoop environment.
2. Complete the tasks for running mappings in the Hadoop environment when Informatica uses Kerberos authentication.
3. Create matching operating system profile user names on the machine that runs the Data Integration Service and each Hadoop cluster node used to run Informatica mapping jobs.
4. Create an Active Directory user that matches the operating system profile user you created in step 3.
5. Create an SPN associated with the user.
   Use the following guidelines when you create the SPN and keytab files:
   - The UPN must be the same as the SPN.
   - Enable delegation in Active Directory.
   - Use the ktpass utility to create an SPN associated with the user and generate the keytabs file.
     For example, enter the following command:

     ```
     ktpass -out infa_hadoop.keytab -mapuser joe -pass tempBG@2008 -princ joe/
     domain12345@HADOOP-AD-REALM -crypto all
     ```

     The `-out` parameter specifies the name and path of the keytab file. The `-mapuser` parameter is the user to which the SPN is associated. The `-pass` parameter is the password for the SPN in the generated keytab. The `-princ` parameter is the SPN.

Configure the Analyst Service

To use the Analyst Service with a Hadoop cluster that uses Kerberos authentication, configure the Analyst Service to use the Kerberos ticket that the Data Integration Service uses.

In the Administrator tool, select the Analyst Service. In the **Processes** tab, edit the Advanced Properties to add the following value to the JVM Command Line Options field: `DINFA_HADOOP_DIST_DIR=<Informatica installation directory>/services/shared/hadoop/<hadoop_distribution>`.

CHAPTER 4

Authorization

This chapter includes the following topics:

Authorization Overview

Authorization controls what a user can do on a Hadoop cluster. For example, a user must be authorized to submit jobs to the Hadoop cluster.

You can use the following systems to manage authorization for Data Engineering Integration:

HDFS permissions

By default, Hadoop uses HDFS permissions to determine what a user can do to a file or directory on HDFS. Additionally, Hadoop implements transparent data encryption in HDFS directories.

Apache Sentry

Sentry is a security plug-in that you can use to enforce role-based authorization for data and metadata on a Hadoop cluster. You can enable high availability for Sentry in the Hadoop cluster. Sentry can secure data and metadata at the table and column level. For example, Sentry can restrict access to columns that contain sensitive data and prevent unauthorized users from accessing the data.

Apache Ranger

Ranger is a security plug-in that you can use to authenticate users of a Hadoop cluster. Ranger manages access to files, folders, databases, tables, and columns. When you perform an action, Ranger verifies that the user meets the policy requirements and has the correct permissions on HDFS. You can enable high availability for Ranger in the Hadoop cluster.

Fine Grained Authorization

Fine grained authorization enables database administrators to impose column-level authorization on Hive tables and views. A more fine-grained level of authorization enables administrators to impose row and column level authorization. You can configure a Hive connection to observe fine grained authorization.

Support for Authorization Systems on Hadoop

Depending on the run-time engine that you use, you can run 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 run-time engines supported for the security management system installed on the Hadoop platform:

Hadoop Distribution	Apache Ranger	Apache Sentry	HDFS Transparent Encryption	SSL/TLS	SQL Authorization
Amazon EMR	No support	No support	No support	No support	No support
Azure HDInsight	- Native - Blaze - Spark	No support	No support	No support	- Native - Blaze - Spark
Cloudera CDH	No support	- Native - Blaze - Spark	- Native - Blaze - Spark	- Native - Blaze - Spark	- Native - Blaze - Spark
Cloudera CDP	- Native - Blaze* - Spark	No support	- Native - Spark	- Native - Spark	- Native - Spark
Hortonworks HDP	- Native - Blaze - Spark	No support	- Native - Blaze - Spark	- Native - Blaze - Spark	- Native - Blaze - Spark
MapR	No support	No support	No support	- Native - Blaze - Spark	No support

** For Cloudera CDP Public Cloud, Apache Ranger is supported on Data Lake clusters only. Data Hub clusters do not support Apache Ranger authorization.*

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 Kerberos for authentication and Apache Ranger for authorization.
- Sqoop cannot access SSL-enabled databases.
- Data Engineering Integration 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.

HDFS Permissions

HDFS permissions determine what a user can do to files and directories stored in HDFS. To access a file or directory, a user must have permission or belong to a group that has permission.

HDFS permissions are similar to permissions for UNIX or Linux systems. For example, a user requires the *r* permission to read a file and the *w* permission to write a file.

When a user or application attempts to perform an action, HDFS checks if the user has permission or belongs to a group with permission to perform that action on a specific file or directory.

SQL Authorization for Hive

SQL standards-based authorization enables database administrators to impose fine grained authorization on Hive tables and views when you read data from a Hive source or a target.

Informatica supports fine grained authorization for Hive sources with Blaze engine, and Hive sources and targets with Spark engines. You can use the Ranger authorization plug-in when you enable fine grained authorization for mappings that run on a Hortonworks HDP cluster or a Cloudera CDP cluster.

You can use the Sentry authorization plug-in when you enable fine grained authorization for mappings that run on a Cloudera CDH cluster. When the mapping accesses Hive sources in Blaze engine and Hive sources and targets in Spark engine on a cluster that uses Sentry authorization and runs in native mode, you can use SQL authorization on the column level if you configure hive.server2.proxy.user in the Hive JDBC connect string.

In this case, the mapping uses the hive.server2.proxy.user value to access Hive sources and targets. When you also configure the mappingImpersonationUserName property, then the mapping uses the mappingImpersonationUserName value to access Hive sources and targets.

You can configure a Hive connection to observe fine grained authorization.

Key Management Servers

Key Management Server (KMS) is an open source key management service that supports HDFS data at rest encryption. You can use the cluster administration utility to configure the KMS for Informatica user access.

You can use the following key management servers to encrypt the data at rest:

- Apache Ranger KMS. Ranger Key Management Store is an open source, scalable cryptographic key management service that supports HDFS data at rest encryption.
- Cloudera Java KMS. For Cloudera CDH clusters, Cloudera provides a Key Management Server based on the Hadoop KeyProvider API to support HDFS data at rest encryption.
- Cloudera Navigator Encrypt. Cloudera Navigator Encrypt is a Cloudera proprietary key management service that secures the data and implements HDFS data at rest encryption.

KMS enables the following functions:

Key management

You can create, update, or delete encryption key zones that control access to functionality.

Access control policies

You can administer access control policies for encryption keys. You can create or edit keys to control access by users to functionality.

Configuring KMS for Informatica User Access

If you use a KMS to encrypt HDFS data at rest, use the cluster administration utility to configure the KMS for Informatica user access.

1. Create a KMS user account for the Informatica user. Add the Informatica user to a new KMS repository, or to an existing KMS repository.

   The user corresponds to the Data Integration Service user or the Kerberos SPN user.
2. Grant permissions to the Informatica user.
3. Create and configure an encryption key.
4. Create an encryption zone that uses the encryption key you created.

   For example:
   ```
   hdfs dfs -mkdir /zone_encr_infa
   hdfs crypto -createZone -keyName infa_key -path /zone_encr_infa
   ```
5. Browse to the Custom KMS Site page and add the following properties:
   ```
   hadoop.kms.proxyuser.<user>.groups=*
   hadoop.kms.proxyuser.<user>.hosts=*
   hadoop.kms.proxyuser.<user>.users=*
   ```
   where <user> is the Informatica user name you configured in Step 1.
6. Update the following properties:
   ```
   hadoop.kms.proxyuser.<user>.hosts
   hadoop.kms.proxyuser.<user>.groups
   ```
7. Search for *proxyuser* in the KMS Configurations area. To register all Hadoop system users with the KMS, add the following properties:
   ```
   hadoop.kms.proxyuser.HTTP.hosts=*
   hadoop.kms.proxyuser.HTTP.users=*
   hadoop.kms.proxyuser.hive.hosts=*
   hadoop.kms.proxyuser.hive.users=*
   hadoop.kms.proxyuser.keyadmin.hosts=*
   hadoop.kms.proxyuser.keyadmin.users=*
   hadoop.kms.proxyuser.nn.hosts=*
   hadoop.kms.proxyuser.nn.users=*
   hadoop.kms.proxyuser.rm.hosts=*
   hadoop.kms.proxyuser.rm.users=*
   hadoop.kms.proxyuser.yarn.hosts=*
   hadoop.kms.proxyuser.yarn.users=*
   ```
8. To use Ranger KMS authorization with a Cloudera CDP cluster, configure an HDFS policy in Ranger. Add the external HDFS location to the Resource path. Assign Read, Write, and Execute permissions for the impersonation user and the Hive user.

Configuring Access to an SSL/TLS-Enabled Cluster

When you use an SSL-enabled or TLS-enabled cluster, you must configure the Informatica domain to communicate with the secure cluster.

Based on the cluster distribution that uses SSL, you perform the following tasks:

Amazon EMR cluster uses SSL/TLS

Import security certificates from the cluster to the Informatica domain. If you created a Hive or S3 connection object manually, configure the connection string properties to access the SSL-enabled cluster.

Cloudera CDH, Cloudera CDP, Dataproc, or Hortonworks HDP cluster uses SSL

Import security certificates from the cluster to the Informatica domain. If you created a Hive connection manually, configure the connection string properties to access the SSL-enabled cluster.

MapR cluster uses SSL

Make sure that the MapR client is configured to communicate with a secure cluster. If you created a Hive connection object manually, configure the connection string properties to access the SSL-enabled cluster.

Configure the Hive Connection for SSL-Enabled Clusters

If you created the Hive connection when you created cluster configurations, the cluster configuration creation wizard enables access to a cluster that uses SSL. If you manually created a Hive connection, you must configure the connection string properties to enable access to a cluster that uses SSL.

If you manually created a Hive connection, add the following property-value pair to the metadata connection string and data access connection string properties:

```
ssl=true
```

For example:

```
jdbc:hive2://<hostname>:<port>/<db>;ssl=true
```

Note: Insert the `ssl=true` flag before the kerberos principal element when you create the Hive connection manually.

Import Security Certificates from an SSL-Enabled Cluster

When you use custom, special, or self-signed security certificates to secure the Hadoop cluster, Informatica services that connect to the cluster require these certificates to be present on the machines that run the application services. Use the keytool utility to import certificates from the cluster.

For more information about the keytool utility, refer to the Oracle documentation.

If a cluster uses SSL, perform the following steps to import security certificates from the cluster to the Data Integration Service and Metadata Access Service machines:

1. Run the following keytool -exportcert command on the cluster to export the certificates:

   ```
   keytool -exportcert
   -alias <alias name>
   -keystore <custom.truststore file location>
   -file <exported certificate file location>
   -storepass <password>
   ```

   Where:

   - -alias specifies the alias name associated with the truststore file.
   - -keystore specifies the location of the truststore file on the cluster.
   - -file specifies the file name and location for the exported certificate file.
   - -storepass specifies the password for the keystore on the cluster.

   The keytool -exportcert command produces a certificate file associated with the alias.

2. Run the following keytool -importcert command on one Data Integration Service machine to import the security certificates:

    ```
    keytool -importcert -trustcacerts
    -alias <alias name>
    -file <exported certificate file location>
    -keystore <java cacerts location>
    -storepass <password>
    ```

    Where:

    - -alias specifies the alias name associated with the certificate file.
    - -file specifies the file name and location of the exported certificate file.
    - -keystore specifies the location of the truststore file on the domain.
    - -storepass specifies the password for the keystore on the domain.

    Important: Import the certificate files one time and then copy them to all machines that host the Data Integration Service and Metadata Access Service. If the Data Integration Service runs on a grid, mappings that you push to the Hadoop environment can fail with initialization errors due to inconsistent binary hex values.

    Depending on whether the Informatica domain uses SSL, you specify the keystore location as follows:

    - If the domain is SSL-enabled, import the certificate file to the following location:
      `<Informatica installation directory>/services/shared/security/infa_truststore.jks`
    - If the domain is not SSL-enabled, import the certificate file to the following location:
      `<Informatica installation directory>/java/jre/lib/security/cacerts`

    The keytool -importcert command imports the security certificates to the keystore location you specify.

Example. Import Security Certificates

The environment includes a Cloudera CDH cluster that uses SSL and an Informatica domain that does not use SSL. You export the security certificate for the user bigdata_user1 from the custom.keystore on the Cloudera CDH cluster to the file exported.cer. Then, you import the export.cer certificate file to the Informatica domain location.

1. Run the following export command:

    ```
    keytool -exportcert -alias bigdata_user1 -keystore ~/custom.truststore -file ~/
    exported.cer
    ```

2. Run the following import command on the Data Integration Service machine:

    ```
    keytool -importcert -alias bigdata_user1 -file ~/exported.cer -keystore <Informatica
    installation directory>/java/jre/lib/security/cacerts
    ```

3. Copy the certificate file to all other machines that host the Data Integration Service and the Metadata Access Service.

Rules and Guidelines for Importing Security Certificates from an SSL-Enabled Cluster

Consider the following rules and guidelines when you import security certificates from an SSL-enabled cluster:

- If a MapR cluster is SSL-enabled, you do not have to import the security certificates. Make sure that the MapR client on the Data Integration Service and Metadata Access Service machines is configured to access an SSL-enabled cluster.
- If a Cloudera CDP cluster is Auto-TLS enabled, import the security certificates before you import the cluster configuration into the domain.

Import Security Certificates from a TLS-Enabled Domain

When the domain is configured to use TLS, you must import the certificates to the default or custom truststore file that the Informatica domain uses.

Default truststore file

If the domain is TLS-enabled and the cluster uses server managed keys, you must import the Baltimore CyberTrust Root certificate to the default truststore file.

Use the keytool utility to import the security certificate.

The default truststore file is located in the following directory: `<Informatica installation home>/services/shared/security/infa_truststore.jks`

Custom truststore file

If the domain is TLS-enabled with a custom truststore file, and the cluster uses server managed keys, get the custom truststore file location from Informatica Administrator, and then import the Baltimore CyberTrust Root certificate to the custom truststore file.

Use the keytool utility to import the security certificate.

To get the custom truststore file location, perform the following steps:

1. In the Administrator tool, click the Manage tab.
2. Click the Services and Nodes view.
3. In the Domain Navigator, select the domain.
4. Get the custom truststore file location from the domain properties.

You can download the Baltimore CyberTrust Root certificates from https://www.digicert.com/digicert-root-certificates.htm.

For more information about downloading the certificates, see https://docs.microsoft.com/en-us/azure/java-add-certificate-ca-store.

Configuring Access to an SSL-Enabled Database

When you use a JDBC connection, you can configure Sqoop connectivity to SSL-enabled databases.

You can include a SSL-enabled JDBC-compliant database as a Sqoop source or target in an Informatica mapping that runs in a Hadoop environment.

For example, you can include the following sources or targets from SSL-enabled JDBC-compliant databases in a Sqoop mapping:

- Microsoft SQL Server
- Oracle

Configure the JDBC Connection for SSL-Enabled Databases

You must configure the Sqoop properties in the JDBC connection to connect to SSL-enabled databases such as Microsoft SQL Server and Oracle.

When you configure a JDBC connection, add the appropriate JDBC connection string for the SSL-enabled database that you want to connect to in the JDBC connection properties. Specify the JDBC connection string in the `connect` Sqoop argument which Sqoop must use to connect to the SSL database.

To configure Sqoop connectivity to an SSL-enabled Oracle database, you must additionally create a parameter file with the required SSL properties. Provide the location of the file through the `--connection-param-file <parameter_file_location>` Sqoop argument in the JDBC connection.

Configuring Sqoop Connectivity to an SSL-Enabled Oracle Database

The secure Oracle database that you want to connect to must use the PKCS12 wallet with the keystore and truststore, TLS_RSA_WITH_AES_256_CBC_SHA cipher algorithm, and TLS version 1.2.

To configure Sqoop connectivity to the Oracle database, add the dependant jars for using Oracle wallets and enable the Oracle PKI provider. Create a parameter file with the SSL properties required to connect to the secure Oracle database and specify the location of this file through the Sqoop argument in the JDBC connection.

1. Create a parameter file and include the following SSL properties in the file:

   ```
   oracle.jdbc.J2EE13Compliant=true
   javax.net.ssl.trustStore=/tmp/truststore.p12
   javax.net.ssl.trustStoreType=PKCS12
   javax.net.ssl.trustStorePassword=informatica
   javax.net.ssl.keyStore=/tmp/ewallet.p12
   javax.net.ssl.keyStoreType=PKCS12
   javax.net.ssl.keyStorePassword=oracle4u
   ```

2. Place the parameter file in a common path in the Data Integration Service machine and in all the nodes of the Hadoop cluster. For example, `/tmp/param_file`
3. Open the `java.security` file in the following location of the Data Integration Service machine and in all the nodes in the Hadoop cluster: `<JAVA_HOME>/jre/lib/security`
4. To enable the Oracle PKI provider, add the following property at position 3 in the `java.security` file:

   `security.provider.3=oracle.security.pki.OraclePKIProvider`

   When you add the property at position 3, the rest of the existing properties from position 3 shifts to subsequent positions.
5. Copy the following dependent .jar files to use the Oracle wallet from the Oracle web site:

   `ojdbc*.jar`

   `oraclepki.jar`

   `osdt_cert.jar`

   `osdt_core.jar`
6. Paste the .jar files to the following directory on the machine where the Data Integration Service runs: `<Informatica installation directory>/externaljdbcjars`
7. Paste the following .jar files at: `<Informatica installation directory>/java/jre/lib/ext/`

   `oraclepki.jar`

```
osdt_cert.jar
```

```
osdt_core.jar
```

8. In the JDBC connection, provide the connection string for the JDBC driver:

   Sample connection string for DataDirect Oracle JDBC driver:

   ```
   jdbc:informatica:oracle://<host_name>:<port>;CatalogOptions=6;
   ServiceName=<service_name>;
   trustStorePassword=<truststore_password>;
   keyStorePassword=<keystore_password>;CryptoProtocolVersion=TLSv1.2;
   keyStore=<keystore_location_of_ewallet.p12_file>;
   trustStore=<truststore_location_of_truststore.p12_file>;
   HostNameInCertificate=<database_host_name>;encryptionMethod=SSL;
   ValidateServerCertificate=True;
   ```

   Sample connection string for Oracle JDBC driver:

   ```
   odbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)
   (HOST=<host>)(PORT=<port_number>))
   (CONNECT_DATA=(SERVICE_NAME=<service_name>)))"
   ```

9. Specify the following Sqoop argument in the JDBC connection to connect to an SSL-enabled Oracle database:

   ```
   --connect jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=<host>)
   (PORT=<port_number>))
   (CONNECT_DATA=(SERVICE_NAME=<service_name>)))"
   ```

10. Specify the following Sqoop argument in the JDBC connection to use the SSL properties defined in parameter file:

    ```
    --connection-param-file <parameter_file_location>,
    ```

    where `parameter_file_location` is the path of the configured parameter file that contains the SSL properties:

11. Recycle the Data Integration Service.

CHAPTER 5

Cluster Configuration

This chapter includes the following topics:

Cluster Configuration Overview

A cluster configuration is an object in the domain that contains configuration information about the compute cluster. The cluster configuration enables the Data Integration Service to push mapping logic to the non-native environment. Import configuration properties from the compute cluster to create a cluster configuration. You can import directly from the cluster or from an archive file.

When you create a cluster configuration associated with the Hadoop environment, the **Cluster Configuration** wizard can create Hadoop, HBase, HDFS, and Hive connections to access the Hadoop environment. When you create a cluster configuration associated with the Databricks environment, the **Cluster Configuration** wizard can create the Databricks connection. If you choose to create the connections, the wizard also associates the configuration object with the connections.

You can override the property values, and you can create user-defined properties based on your requirements. When property values change on the cluster, you can refresh the cluster configuration, either directly from the cluster or from an archive file.

The cluster configuration contains properties for the cluster type and version. You can edit the version property to any supported version. After you change the version, you must restart the Data Integration Service.

Consider the following high-level process to manage cluster configurations:

1. Import the cluster configuration, choosing to create connections that require the cluster configuration.
2. Edit the cluster configuration. Override imported property values and add user-defined properties.

3. Refresh the cluster configuration. When property values change on the cluster, refresh the cluster configuration to import the changes.

Cluster Configuration and Connections

When you create a cluster configuration, you can choose to create connections. All cluster type connections used to run a mapping must be associated with a cluster configuration.

If you choose to create connections, the **Cluster Configuration** wizard associates the cluster configuration with each connection that it creates.

The wizard creates the following connections when you create a cluster configuration associated with the Hadoop environment:

- Hive
- HBase
- Hadoop
- HDFS

The wizard creates a Databricks connection when you create a cluster configuration associated with the Databricks environment.

The wizard uses the following naming convention when it creates connections: <connection type>_<cluster configuration name>, such as Hive_ccMapR.

If you do not choose to create connections, you must manually create them and associate the cluster configuration with them.

Copying a Connection to Another Domain

When you copy a connection associated with a cluster configuration to another domain, you must first create a cluster configuration in the target domain.

Create a cluster configuration with the same name as the one that is associated with the connection in the source domain.

Note: When you create a cluster configuration in the domain, use the **Cluster Configuration** wizard. Informatica does not recommend importing the archive file into the domain to create a cluster configuration. If you import the archive file into the domain, the user-defined properties are converted to imported properties. When you subsequently refresh the cluster configuration, the refresh operation replaces the values of properties with cluster property values, and removes properties that do not exist on the cluster.

1. Identify a connection to copy, and note the name of the cluster configuration that is associated with it.
2. In the target domain, create a cluster configuration of the same name.
3. Choose not to create connections with creation of the cluster configuration.
4. Copy the connection to the target domain.

The connection has an associated cluster configuration.

Cluster Configuration Views

You can manage cluster configurations on the **Connections** tab of the Administrator tool.

When you highlight a cluster configuration on the **Connections** tab, you can view the cluster configuration details on the right pane. The cluster configuration displays the following components:

Active Properties view

Displays general properties and all run-time properties and values. You can view the following properties:

- Imported properties with imported values and any overridden values.
- User-defined properties and values. User-defined property values appear as overridden values.

Overridden Properties view

Displays only properties with overridden values, including imported properties and user-defined properties.

Permissions view

Configure permissions to perform actions on cluster configurations.

Actions menu

From the Actions menu, you can perform the following actions:

- Refresh the cluster configuration from the Hadoop cluster.
- Export the configuration to an archive file, required by the Developer tool to access cluster metadata at design-time.

You can create, edit, and delete properties from the **Active Properties** view and the **Overridden Properties** view.

Note: The **Active Properties** and **Overridden Properties** views display configuration sets with names based on the associated *-site.xml file on the cluster. For example, the properties from the cluster core-site.xml file appear under the configuration set name core_site_xml.

Active Properties View

The **Active Properties** view displays all cluster properties, both imported and user-defined.

The **Active Properties** view contains general properties and all run-time properties. General properties include the cluster configuration name, ID, description, distribution type, and the last date of refresh. Run-time properties are organized into configuration sets based on the corresponding *-site.xml files on the cluster. For example, the hive-site.xml configuration set contains all of the properties and values imported from the hive-site.xml file on the cluster.

The cluster configuration can contain the following types of run-time properties:

Imported properties

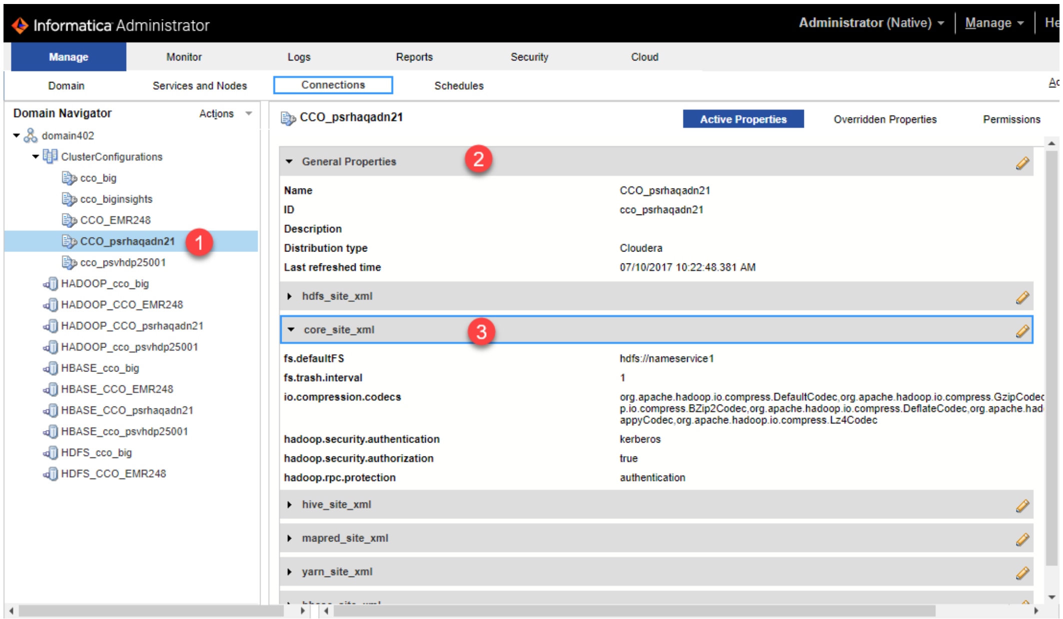
Properties and values imported from the cluster or file. You can override property values based on your requirements. Some cluster configuration properties contain sensitive information, such as passwords. The Service Manager masks the value of sensitive properties with asterisk characters when you import or refresh the cluster configuration. The masked values appear in the Administrator tool and in infacmd results.

User-defined properties

You can create user-defined properties based on processing requirements. When you create a user-defined property, the value appears as an overridden value.

Active properties are properties that the Data Integration Service uses at run time. Each expanded configuration set of the **Active Properties** view displays these active values. If a property has an overridden value, the Data Integration Service uses the overridden value as the active value. If the property does not have an overridden value, the Data Integration Service uses the imported value as the active value. To see the imported value of a property that is overridden, click the edit icon.

The following image shows cluster configurations in the **Domain Navigator**.



Name CCO_psrhaqadn21
ID cco_psrhaqadn21
Description
Distribution type Cloudera
Last refreshed time 07/10/2017 10:22:48.381 AM

fs.defaultFS hdfs://nameservice1
fs.trash.interval 1
io.compression.codecs org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.GzipCodec
p.io.compress.BZip2Codec,org.apache.hadoop.io.compress.DeflateCodec,org.apache.had
appyCodec,org.apache.hadoop.io.compress.Lz4Codec
hadoop.security.authentication kerberos
hadoop.security.authorization true
hadoop.rpc.protection authentication

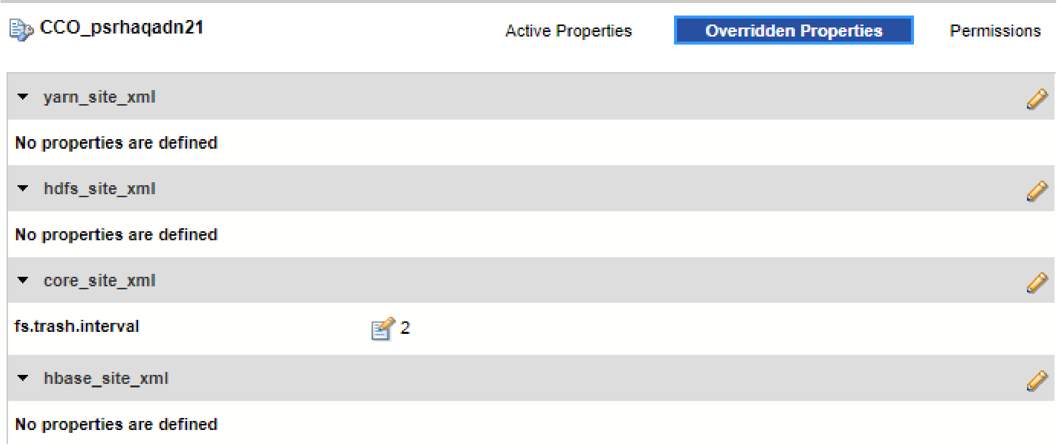
1. The **Cluster Configurations** node in the **Domain Navigator** displays the cluster configurations in the domain.
2. The right pane shows the general properties and configuration sets. The General Properties set is expanded to show general property values.
3. The core-site.xml configuration set is expanded to show the properties that it contains.

Overridden Properties View

The **Overridden Properties** view displays only properties with overridden values.

The **Overridden Properties** view includes user-defined properties and imported properties that you overrode. The values that appear in the view are the active values. To see imported values, click the edit icon.

The following image shows a property in the core-site.xml configuration set with an overridden value of 2:



yarn_site_xml

No properties are defined

hdfs_site_xml

No properties are defined

core_site_xml

fs.trash.interval 2

hbase_site_xml

No properties are defined

Note that configuration sets that do not contain overrides display a message indicating that no properties are defined.

Create a Hadoop Cluster Configuration

Import the Hadoop cluster information into the domain. When you import cluster information, you import values from *-site.xml files to create a domain object called a cluster configuration.

Choose one of the following options to import cluster properties:

Import from cluster

> When you import directly from the cluster, you enter cluster connection information. The Service Manager uses the information to connect to the cluster and get cluster configuration properties.
>
> **Note:** You can import directly from Azure HDInsight, Cloudera CDH, Cloudera CDP, and Hortonworks HDP clusters.

Import from file

> When you import from a file, you browse to an archive file that the Hadoop administrator created. Use this option if the Hadoop administrator requires you to do so.
>
> **Note:** If you import from a MapR or Amazon EMR cluster, you must import from a file.

Before You Import

Before you can import the cluster configuration, you must get information from the Hadoop administrator, based on the method of import.

If you import directly from the cluster, contact the Hadoop administrator to get cluster connection information. If you import from a file, get an archive file of exported cluster information.

For more information about required cluster information, see the *Data Engineering Integration Guide*

Note: To import from Amazon EMR or MapR, you must import from an archive file.

Importing a Hadoop Cluster Configuration from the Cluster

When you import the Hadoop cluster configuration directly from the cluster, you provide information to connect to the cluster.

Get cluster connection information from the Hadoop administrator.

1. From the **Connections** tab, click the **ClusterConfigurations** node in the Domain Navigator.
2. From the Actions menu, select **New** > **Cluster Configuration**.

   The **Cluster Configuration** wizard opens.
3. Configure the following General properties:

Property	Description
Cluster configuration name	Name of the cluster configuration.
Description	Optional description of the cluster configuration.
Distribution type	The cluster Hadoop distribution type.
Method to import the cluster configuration	Choose **Import from cluster**.
Create connections	Choose to create Hadoop, HDFS, Hive, and HBase connections. If you choose to create connections, the Cluster Configuration wizard associates the cluster configuration with each connection that it creates. The Hadoop connection contains default values for properties such as cluster environment variables, cluster path variables, and advanced properties. Based on the cluster environment and the functionality that you use, you can add to the default values or change the default values of these properties. For a list of Hadoop connection properties to configure, see Hadoop Cluster Connection Properties. If you do not choose to create connections, you must manually create them and associate the cluster configuration with them. **Important:** When the wizard creates the Hive connection, it populates the Metadata Connection String and the Data Access Connection String properties with the value from the hive.metastore.uris property. If the Hive metastore and HiveServer2 are running on different nodes, you must update the Metadata Connection String to point to the HiveServer2 host.

   The cluster properties appear.

4. Configure the following properties:

Property	Description
Host	Host name or IP address of the cluster manager.
Port	Port of the cluster manager.
User ID	Cluster user ID.

Property	Description
Password	Password for the user.
Cluster name	Name of the cluster. Use the display name if the cluster manager manages multiple clusters. If you do not provide a cluster name, the wizard imports information based on the default cluster.

5. Click **Next** and verify the cluster configuration information on the summary page.

Importing a Hadoop Cluster Configuration from a File

You can import properties from an archive file to create a cluster configuration.

Before you import from the cluster, you must get the archive file from the Hadoop administrator.

1. From the **Connections** tab, click the **ClusterConfigurations** node in the Domain Navigator.
2. From the Actions menu, select **New** > **Cluster Configuration**.

   The **Cluster Configuration** wizard opens.
3. Configure the following properties:

Property	Description
Cluster configuration name	Name of the cluster configuration.
Description	Optional description of the cluster configuration.
Distribution type	The cluster Hadoop distribution type.
Method to import the cluster configuration	Choose **Import from file** to import properties from an archive file.
Create connections	Choose to create Hadoop, HDFS, Hive, and HBase connections. If you choose to create connections, the Cluster Configuration wizard associates the cluster configuration with each connection that it creates. The Hadoop connection contains default values for properties such as cluster environment variables, cluster path variables, and advanced properties. Based on the cluster environment and the functionality that you use, you can add to the default values or change the default values of these properties. For a list of Hadoop connection properties to configure, see "Configuring Hadoop Connection Properties" on page 149. If you do not choose to create connections, you must manually create them and associate the cluster configuration with them. **Important:** When the wizard creates the Hive connection, it populates the Metadata Connection String and the Data Access Connection String properties with the value from the hive.metastore.uris property. If the Hive metastore and HiveServer2 are running on different nodes, you must update the Metadata Connection String to point to the HiveServer2 host.

4. Click **Browse** to select a file. Select the file and click **Open**.
5. Click **Next** and verify the cluster configuration information on the summary page.

Create a Databricks Cluster Configuration

A Databricks cluster configuration is an object in the domain that contains configuration information about the Databricks cluster. The cluster configuration enables the Data Integration Service to push mapping logic to the Databricks environment.

Use the Administrator tool to import configuration properties from the Databricks cluster to create a cluster configuration. You can import configuration properties from the cluster or from a file that contains cluster properties. You can choose to create a Databricks connection when you perform the import.

Note:

- Ensure that you integrate a Databricks cluster with only one Informatica domain.
- Ensure that a Databricks workspace has access to the database VM where you create the reference table.

Importing a Databricks Cluster Configuration from the Cluster

When you import the cluster configuration directly from the cluster, you provide information to connect to the cluster.

Before you import the cluster configuration, get cluster information from the Databricks administrator.

1. From the **Connections** tab, click the **ClusterConfigurations** node in the Domain Navigator.
2. From the Actions menu, select **New** > **Cluster Configuration**.
   The **Cluster Configuration** wizard opens.
3. Configure the following properties:

Property	Description
Cluster configuration name	Name of the cluster configuration.
Description	Optional description of the cluster configuration.
Distribution type	The distribution type. Choose **Databricks**.
Method to import the cluster configuration	Choose **Import from cluster**.
Databricks domain	Domain name of the Databricks deployment.
Databricks access token	The token ID created within Databricks, 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.

Property	Description
Databricks cluster ID	The cluster ID of the Databricks cluster. To find the cluster ID on the Databricks portal, follow these steps: 1. Select **Clusters** from the object bar on the left side. 2. Select the cluster you want to integrate with the Informatica domain. 3. Click the **Spark ID** tab and expand the list of **Spark Properties**. 4. Select the **Tags** tab.
Create connection	Choose to create a Databricks connection. If you choose to create a connection, the **Cluster Configuration** wizard associates the cluster configuration with the Databricks connection. If you do not choose to create a connection, you must manually create one and associate the cluster configuration with it.

4. Click **Next** to verify the information on the summary page.

Importing a Databricks Cluster Configuration from a File

You can import properties from an archive file to create a cluster configuration.

Complete the following tasks to import a Databricks cluster from a file:

1. Get required cluster properties from the Databricks administrator.
2. Create an .xml file with the cluster properties, and compress it into a .zip or .tar file.
3. Log in to the Administrator tool and import the file.

Create the Import File

To import the cluster configuration from a file, you must create an archive file.

To create the .xml file for import, you must get required information from the Databricks administrator. You can provide any name for the file and store it locally.

The following table describes the properties required to import the cluster information:

Property Name	Description
cluster_name	Name of the Databricks cluster.
cluster_ID	The cluster ID of the Databricks cluster.
baseURL	URL to access the Databricks cluster. This is the domain URL that appears in your browser menu bar. It commonly incorporates your account region. For example, `https://southcentralus.azuredatabricks.net` or `https://westus.azuredatabricks.net`.
accesstoken	The token ID created within Databricks required for authentication.

Optionally, you can include other properties specific to the Databricks environment.

When you complete the .xml file, compress it into a .zip or .tar file for import.

Sample Import File

The following text shows a sample import file with the required properties:

```
<?xml version="1.0" encoding="UTF-8"?><configuration>
  <property>
    <name>cluster_name</name>
    <value>my_cluster</value>
  </property>
  <property>
    <name>cluster_id</name>
    <value>0926-294544-bckt123</value>
  </property>
  <property>
    <name>baseURL</name>
    <value>https://<region>.azuredatabricks.net</value>
  </property>
  <property>
    <name>accesstoken</name>
    <value>dapicf76c2d4567c6sldn654fe875936e778</value>
  </property>
</configuration>
```

Import the Cluster Configuration

After you create the .xml file with the cluster properties, use the Administrator tool to import into the domain and create the cluster configuration.

1. From the **Connections** tab, click the **ClusterConfigurations** node in the Domain Navigator.
2. From the Actions menu, select **New** > **Cluster Configuration**.
   The **Cluster Configuration** wizard opens.
3. Configure the following properties:

Property	Description
Cluster configuration name	Name of the cluster configuration.
Description	Optional description of the cluster configuration.
Distribution type	The distribution type. Choose **Databricks**.
Method to import the cluster configuration	Choose **Import from file**.
Upload configuration archive file	The full path and file name of the file. Click the Browse button to navigate to the file.
Create connection	Choose to create a Databricks connection. If you choose to create a connection, the **Cluster Configuration** wizard associates the cluster configuration with the Databricks connection. If you do not choose to create a connection, you must manually create one and associate the cluster configuration with it.

4. Click **Next** to verify the information on the summary page.

Edit the Cluster Configuration

You can edit property values in a cluster configuration. You can also add user-defined properties, override imported property values, and delete properties within a configuration set.

To edit the cluster configuration, you can access the **Edit** dialog box for a configuration set on the **Active Properties** view or the **Overridden Properties** view. Within the **Edit** dialog box, you can use the filter control to find properties.

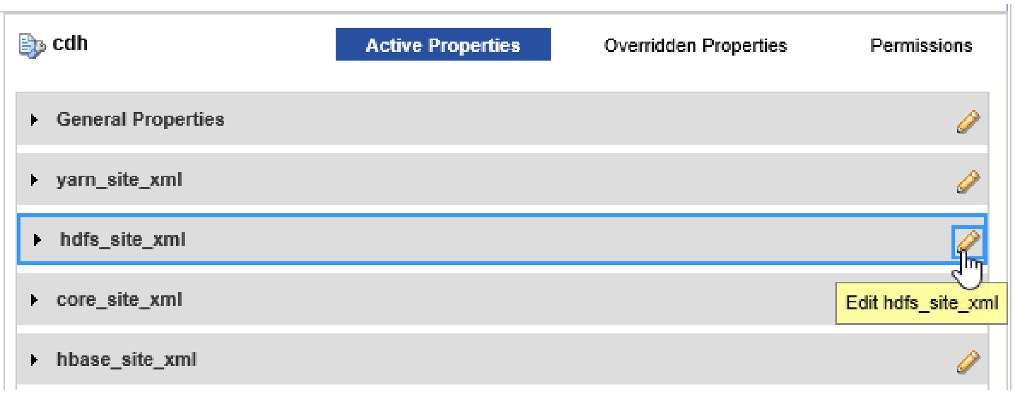
If you edit properties in a cluster configuration that the Data Integration Service used to run a mapping, recycle the Data Integration Service for changes to take effect. For example, if you change the Hadoop distribution version for a cluster configuration that the Data Integration Service used to run a mapping, recycle the Data Integration Service for the change to take effect.

Filtering Cluster Configuration Properties

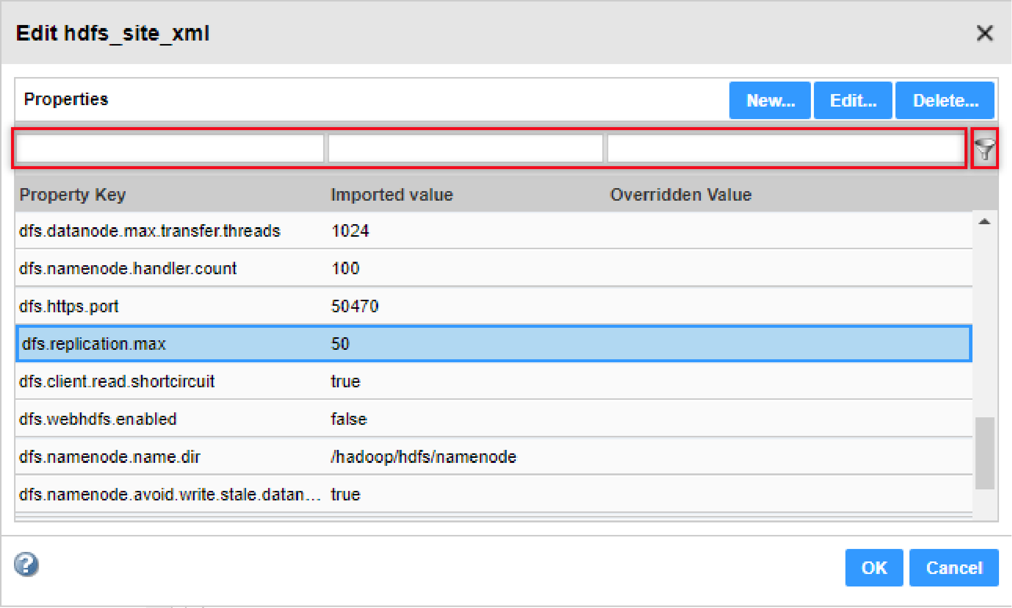
You can search for properties within a configuration set by using the filter controls.

You can filter properties in the **Active Properties** view or the **Overridden Properties** view. You might want to filter properties when a configuration set contains a large number of properties.

1. In the **Active Properties** view or the **Overridden Properties** view, expand a configuration set.
2. Click the **Edit** icon on the name bar of the configuration set that you want to edit.
   The following image shows the **Edit** icon for the hdfs-site.xml configuration set:



3. Enter text in the filter text entry pane above any column, and then click the filter icon. You can search by property, imported value, overridden value.
   The following image shows the filter text entry panes and the filter icon:



Property Key	Imported value	Overridden Value
dfs.datanode.max.transfer.threads	1024	
dfs.namenode.handler.count	100	
dfs.https.port	50470	
dfs.replication.max	50	
dfs.client.read.shortcircuit	true	
dfs.webhdfs.enabled	false	
dfs.namenode.name.dir	/hadoop/hdfs/namenode	
dfs.namenode.avoid.write.stale.datan...	true	

Overriding Imported Properties

You can override property values or you can update overrides from the **Active Properties** view or the **Overridden Properties** view.

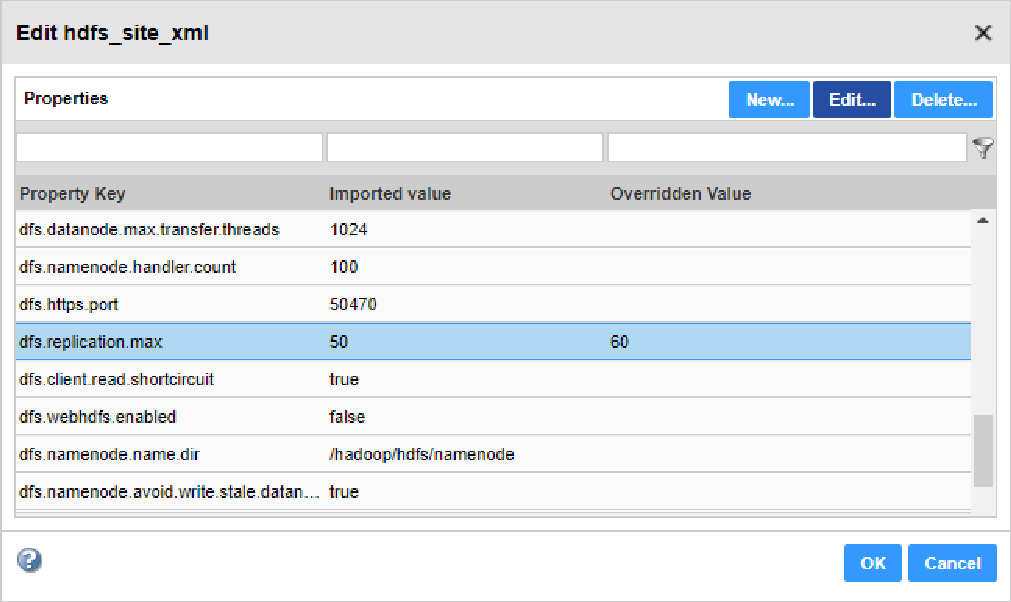
1. Expand the configuration set containing the property that you want to edit.
2. Click the **Edit** icon on the name bar of the configuration set that you want to edit.

   The **Edit** dialog box opens.
3. Optionally, use the filter controls to find a property.
4. Select the property to edit and click **Edit**.

   The **Edit Property** dialog box opens.

5. Enter a value in the **Overridden Value** pane, and click **OK**.

   The following image shows the overridden value in the **Edit** dialog box:



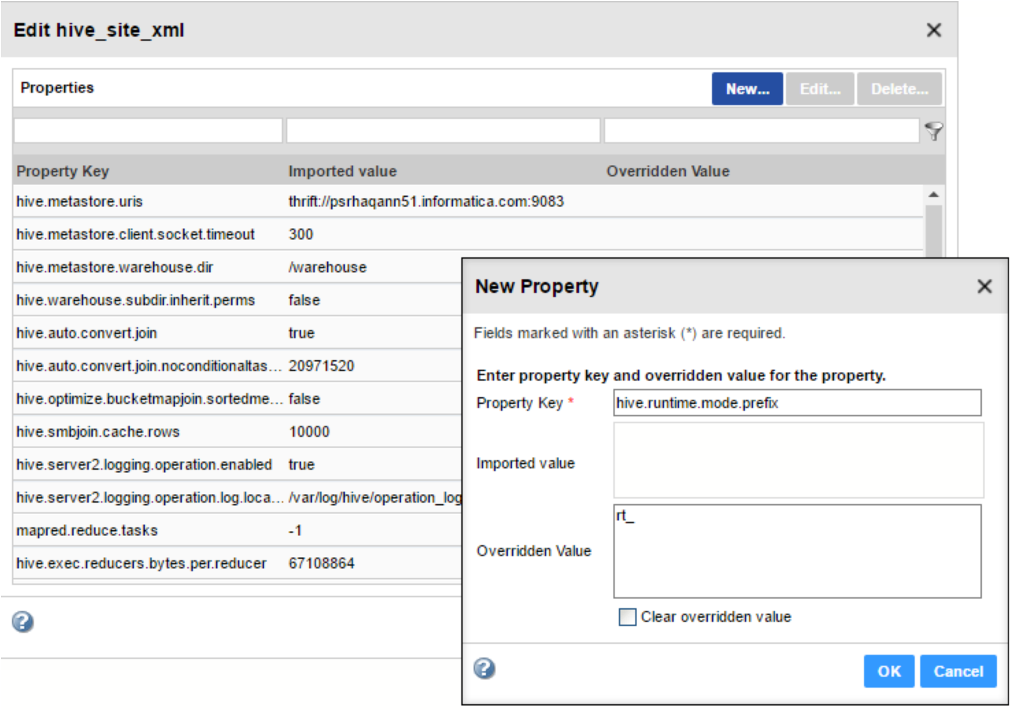
Property Key	Imported value	Overridden Value
dfs.datanode.max.transfer.threads	1024	
dfs.namenode.handler.count	100	
dfs.https.port	50470	
dfs.replication.max	50	60
dfs.client.read.shortcircuit	true	
dfs.webhdfs.enabled	false	
dfs.namenode.name.dir	/hadoop/hdfs/namenode	
dfs.namenode.avoid.write.stale.datan...	true	

Creating User-Defined Properties

You can create user-defined properties in the **Active Properties** view or the **Overridden Properties** view. When you create a user-defined property, you configure an overridden value. You cannot configure an imported value in a user-defined property.

You can create a user-defined property based on your requirements.

1. Expand the configuration set where you want to create a property.
2. Click the **Edit** icon on the name bar of the configuration set that you want to edit.
   The **Edit** dialog box opens.
3. Click **New**.
   The **New Property** dialog box opens.



Property Key	Imported value	Overridden Value
hive.metastore.uris	thrift://psrhaqann51.informatica.com:9083	
hive.metastore.client.socket.timeout	300	
hive.metastore.warehouse.dir	/warehouse	
hive.warehouse.subdir.inherit.perms	false	
hive.auto.convert.join	true	
hive.auto.convert.join.noconditionaltas...	20971520	
hive.optimize.bucketmapjoin.sortedme...	false	
hive.smbjoin.cache.rows	10000	
hive.server2.logging.operation.enabled	true	
hive.server2.logging.operation.log.loca...	/var/log/hive/operation_log	
mapred.reduce.tasks	-1	
hive.exec.reducers.bytes.per.reducer	67108864	

Fields marked with an asterisk (*) are required.

Enter property key and overridden value for the property.

Property Key *: hive.runtime.mode.prefix

Imported value:

Overridden Value: rt_

□ Clear overridden value

4. Configure the following properties:

Property	Description
Property Key	Name of the property that you want to enter.
Overridden Value	The property value. To clear the contents of this field, select **Clear overridden value**.

Important: If you create a property with the same name as a property that exists in a different configuration set, the Data Integration Service might use either property at run time, leading to unpredictable results.

5. Click **OK**.

Deleting Cluster Configuration Properties

You can delete imported and user-defined properties from a configuration set.

1. Select a cluster configuration to edit.
2. In the **Active Properties** view or the **Overridden Properties** view, expand a configuration set.
3. Click the **Edit** icon on the name bar of the configuration set that you want to edit.

   The configuration set expands to show its contents.
4. Optionally, use the filter control at the top of the **Property Key** column to filter the properties.
5. Select the property and click **Delete**.

Note: Imported properties that you delete will be restored if you refresh the cluster configuration.

Refresh the Cluster Configuration

When property values change on the cluster, refresh the cluster configuration to import the changes. Similar to when you import the cluster configuration, you can refresh it either directly or from a .zip or .tar archive file.

You can refresh the cluster configuration from the Actions menu.

The refresh operation updates the cluster configuration in the following ways:

- Unedited values of imported properties are refreshed with the value from the cluster.
- The refresh operation refreshes the contents of the cluster configuration and properties in the connections that are associated with it, such as fs.defaultFS. For example, if you refresh a Hadoop cluster configuration from an archive file that does not contain one of the required *-site.xml files, the refresh cluster configuration drops the configuration set. If a missing *-site.xml file contained properties that get propagated to a connection, the refresh operation succeeds, but the connection refresh fails.
- If you override an imported property value, the refresh operation refreshes the imported value and maintains the overridden value. The Administrator tool displays the updated imported value and the active override value.
- If you override an imported property value, and the property is subsequently removed from the cluster, the refresh operation converts the property to a user-defined property.
- User-defined properties that do not exist in the cluster are not affected.
- If you configure a user-defined property that is subsequently configured on the cluster, the refresh operation converts the user-defined property to an imported property. The user-defined property value becomes an overridden value of the imported property.
- If you deleted an imported property, the refresh operation imports the property again if it exists on the cluster.

Note: If you refresh the cluster configuration while a mapping task is queued, the queued task uses outdated cluster configuration settings instead of the refreshed settings. This might happen when you manually refresh the cluster configuration, or start a task that automatically refreshes the cluster configuration. For best results, verify that all mapping tasks are complete before you refresh the cluster configuration.

Example - Cluster Configuration Refresh

The following example shows the process for importing a property added during a refresh operation after you create it as a user-defined property.

1. You add the following user-defined property to the cluster configuration:

```
<property>
  <name>runtime.engine.prefix</name>
  <value>rt_</value>
  <description>prefixes the runtime output table by this string</description>
</property>
```

2. The Hadoop administrator adds the property to the cluster, but with a different value, as follows:

```
<property>
  <name>runtime.engine.prefix</name>
  <value>runtime_</value>
  <description>prefixes the runtime output table by this string</description>
</property>
```

3. You refresh the cluster configuration, and the refresh operation performs the following tasks.
   a. Converts the user-defined property to an imported property.

b. Maintains the user-defined "rt_" value as an overridden value.

c. Imports the cluster value "runtime_" as the imported value.

Delete a Cluster Configuration

You cannot delete a cluster configuration that has associated connections. You can associate the connections with a different cluster configuration before you delete the cluster configuration.

You can also use the infacmd cluster DeleteConfguration command to delete connections when you delete the cluster configuration.

You must have write permission to delete a cluster configuration, and Manage Connections privilege to delete connections.

To delete a cluster configuration, click the **Actions** menu and select **Delete**.

Cluster Configuration Privileges and Permissions

You manage user security with privileges, roles, and permissions. Privileges determine the actions that users can perform on a cluster configuration. Permissions define the level of access that users and groups have to a cluster configuration.

Privileges and Roles

Cluster configuration privileges and roles determine the actions that users can perform using the Administrator tool and the infacmd command line program.

The following privileges and roles are required to perform certain actions on the cluster configuration:

Domain Administration privilege group

A user assigned the Administrator role for the domain can configure cluster configurations.

Manage Connections privilege

Users or groups assigned the Manage Connections privilege can create, refresh, and delete cluster configurations. Users can also set and clear configuration properties.

Permissions

Permissions control the level of access that a user or group has for a cluster configuration.

You can configure permissions for a cluster configuration in the Administrator tool and using infacmd.

Any cluster configuration permission that is assigned to a user or group in one tool also applies in the other tool. For example, you grant GroupA permission on ConfigurationA using the Informatica command line interface. GroupA has permission on ConfigurationA in the Developer tool also.

The following Informatica components use the cluster configuration permissions:

- Administrator tool. Enforces read, write, execute, and grant permissions on cluster configurations.

- Informatica command line interface. Enforces read, write, execute, and grant permissions on cluster configurations.
- Developer tool. Enforces read, write, and execute permissions on cluster configurations.
- Data Integration Service. Enforces execute permissions when a user tries to preview data or run a mapping, scorecard, or profile.

Types of Cluster Configuration Permissions

You can assign different permission types to users to perform the following actions:

Permission Type	Action
Read	View the cluster configuration.
Write	Edit and refresh the cluster configuration. Set and clear configuration properties. Export the cluster configuration with sensitive properties. Delete the cluster configuration. Users with write permission inherit read permission.
Execute	Run mappings in the Hadoop environment.
Grant	Grant permission on the cluster configuration to other users and groups. Users with grant permission inherit read permission.
All	Inherit read, write, execute, and grant permissions.
None	Remove permissions for the user.

Note the following default permissions for cluster configurations:

- The domain administrator has all permissions on all cluster configurations.
- The user that creates a cluster configuration has read, write, execute, and grant permission for the cluster configuration.
- All users have permission to view the cluster configuration name.

CHAPTER 6

Cloud Provisioning Configuration

This chapter includes the following topics:

Cloud Provisioning Configuration Overview

A cloud provisioning configuration is an object in the domain that contains information about the cloud platform. The cloud provisioning configuration gives the Data Integration Service the information it needs to create a cluster on the cloud platform.

Create a cloud provisioning configuration when you configure a cluster workflow. You create a cluster workflow to automate the creation of clusters and workflow tasks on an Amazon Web Services, Microsoft Azure, or the Databricks compute cluster.

The Data Integration Service uses the information in the cloud provisioning configuration to establish a relationship between the workflow Create Cluster task and the cloud platform, and to run tasks on the cluster that the workflow creates. Using authentication credentials from the cloud provisioning configuration, the Data Integration Service submits jobs to the compute cluster using the REST API.

The cluster connection that the cluster workflow uses contains a reference to the cloud provisioning configuration.

Consider the following high-level process for using the cloud provisioning connection:

1. Verify prerequisites.
2. Create the cloud provisioning configuration.
3. Create a cluster connection for the workflow.

After you create the cloud provisioning configuration and the cluster connection, a developer uses the Developer tool to create, deploy, and run a cluster workflow.

Verify Prerequisites

Verify the following cloud platform and domain prerequisites.

Cloud Platform Prerequisites

Verify the following prerequisites for the cloud platform:

- Create a user account with administrator permissions on the cloud platform.
- Create a resource on the cloud platform where you can create clusters.
  - On AWS, create a Virtual Private Cloud (VPC)
  - On Azure, create a Virtual Network (vnet)

  The Informatica domain and the cluster the workflow creates must be located on this resource.
- If the Informatica domain is installed on-premises, enable DNS resolution.

Domain Prerequisites

Verify the following prerequisites for the Informatica domain:

- An Informatica domain must be installed. The domain can reside on an instance on the Amazon or Microsoft Azure cloud platform, or on an on-premises machine. If the Informatica instance is installed on-premises, you must configure the VPN to connect to the AWS VPC or Azure vnet where the cluster runs.
- You must have permissions to create connections on the domain.
- To create clusters on AWS, the AWS administrator must open the required ports for the security group to use in the VPC where you want to create the cluster.

Enable DNS Resolution from an On-Premises Informatica Domain

If the Informatica domain is installed on-premises, you must enable DNS resolution over the VPN that connects the domain and the cloud platform.

You can enable DNS resolution on Amazon AWS or Microsoft Azure.

Amazon AWS

To run mappings from an on-premises deployment of Data Engineering Integration to a cluster on AWS, you must install and configure Unbound on an EC2 instance. Unbound enables DNS resolution from an on-premises network. To read how to install and configure Unbound in the AWS VPC, see the AWS documentation.

Microsoft Azure

To run mappings from an on-premises deployment of Data Engineering Integration to a cluster on Azure, you must use the Bind utility on the Azure virtual network.

Follow the steps in the Microsoft Azure article "DNS Configuration."

The article gives an example of the contents of the `/etc/bind/named.conf.options` file. You can put a list of available IP addresses on the domain network in the goodclients portion of the file. The following excerpt shows an example:

```
//Add the IP range of the joined network to this list
acl goodclients {
    1.2.3.0/24; # IP address range of the virtual network
    1.2.4.0/24;
    1.2.5.0/24;
    1.2.6.0/24;
    1.2.3.253;
    1.2.3.254;
```

```
        localhost;
        localnets;
    };
```

AWS Cloud Provisioning Configuration Properties

The properties in the AWS cloud provisioning configuration enable the Data Integration Service to contact and create resources on the AWS cloud platform.

General Properties

The following table describes cloud provisioning configuration general properties:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description.	Optional. Description of the cloud provisioning configuration.
AWS Access Key ID	Optional. ID of the AWS access key, which AWS uses to control REST or HTTP query protocol requests to AWS service APIs. If you do not specify a value, Informatica attempts to follow the Default Credential Provider Chain.
AWS Secret Access Key	Secret component of the AWS access key. Required if you specify the AWS Access Key ID.
Region	Region in which to create the cluster. This must be the region in which the VPC is running. Use AWS region values. For a list of acceptable values, see AWS documentation. **Note:** The region where you want to create the cluster can be different from the region in which the Informatica domain is installed.

Permissions

The following table describes cloud provisioning configuration permissions properties:

Property	Description
EMR Role	Name of the service role for the EMR cluster that you create. The role must have sufficient permissions to create a cluster, access S3 resources, and run jobs on the cluster. When the AWS administrator creates this role, they select the "EMR" role. This contains the default AmazonElasticMapReduceRole policy. You can edit the services in this policy.
EC2 Instance Profile	Name of the EC2 instance profile role that controls permissions on processes that run on the cluster. When the AWS administrator creates this role, they select the "EMR Role for EC2" role. This includes S3 access by default.
Auto Scaling Role	Required if you configure auto-scaling for the EMR cluster. This role is created when the AWS administrator configures auto-scaling on any cluster in the VPC. Default: When you leave this field blank, it is equivalent to setting the Auto Scaling role to "Proceed without role" when the AWS administrator creates a cluster in the AWS console.

EC2 Configuration

The following table describes cloud provisioning configuration EC2 configuration properties:

Property	Description
EC2 Key Pair	EC2 key pair to enable communication with the EMR cluster master node. Optional. This credential enables you to log into the cluster. Configure this property if you intend the cluster to be non-ephemeral.
EC2 Subnet	ID of the subnet on the VPC in which to create the cluster. Use the subnet ID of the EC2 instance where the cluster runs.
Master Security Group	Optional. ID of the security group for the cluster master node. Acts as a virtual firewall to control inbound and outbound traffic to cluster nodes. Security groups are created when the AWS administrator creates and configures a cluster in a VPC. In the AWS console, the property is equivalent to ElasticMapReduce-master. You can use existing security groups, or the AWS administrator might create dedicated security groups for the ephemeral cluster. If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Master Security Groups	Optional. IDs of additional security groups to attach to the cluster master node. Use a comma-separated list of security group IDs.
Core and Task Security Group	Optional. ID of the security group for the cluster core and task nodes. When the AWS administrator creates and configures a cluster In the AWS console, the property is equivalent to the ElasticMapReduce-slave security group If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Core and Task Security Groups	Optional. IDs of additional security groups to attach to cluster core and task nodes. Use a comma-separated list of security group IDs.
Service Access Security Group	EMR managed security group for service access. Required when you provision an EMR cluster in a private subnet.

General Properties

The following table describes cloud provisioning configuration general properties:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description.	Optional. Description of the cloud provisioning configuration.
AWS Access Key ID	Optional. ID of the AWS access key, which AWS uses to control REST or HTTP query protocol requests to AWS service APIs. If you do not specify a value, Informatica attempts to follow the Default Credential Provider Chain.
AWS Secret Access Key	Secret component of the AWS access key. Required if you specify the AWS Access Key ID.
Region	Region in which to create the cluster. This must be the region in which the VPC is running. Use AWS region values. For a list of acceptable values, see AWS documentation. **Note:** The region where you want to create the cluster can be different from the region in which the Informatica domain is installed.

Permissions

The following table describes cloud provisioning configuration permissions properties:

Property	Description
EMR Role	Name of the service role for the EMR cluster that you create. The role must have sufficient permissions to create a cluster, access S3 resources, and run jobs on the cluster. When the AWS administrator creates this role, they select the “EMR” role. This contains the default AmazonElasticMapReduceRole policy. You can edit the services in this policy.
EC2 Instance Profile	Name of the EC2 instance profile role that controls permissions on processes that run on the cluster. When the AWS administrator creates this role, they select the “EMR Role for EC2” role. This includes S3 access by default.
Auto Scaling Role	Required if you configure auto-scaling for the EMR cluster. This role is created when the AWS administrator configures auto-scaling on any cluster in the VPC. Default: When you leave this field blank, it is equivalent to setting the Auto Scaling role to “Proceed without role” when the AWS administrator creates a cluster in the AWS console.

EC2 Configuration

The following table describes cloud provisioning configuration EC2 configuration properties:

Property	Description
EC2 Key Pair	EC2 key pair to enable communication with the EMR cluster master node. Optional. This credential enables you to log into the cluster. Configure this property if you intend the cluster to be non-ephemeral.
EC2 Subnet	ID of the subnet on the VPC in which to create the cluster. Use the subnet ID of the EC2 instance where the cluster runs.
Master Security Group	Optional. ID of the security group for the cluster master node. Acts as a virtual firewall to control inbound and outbound traffic to cluster nodes. Security groups are created when the AWS administrator creates and configures a cluster in a VPC. In the AWS console, the property is equivalent to ElasticMapReduce-master. You can use existing security groups, or the AWS administrator might create dedicated security groups for the ephemeral cluster. If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Master Security Groups	Optional. IDs of additional security groups to attach to the cluster master node. Use a comma-separated list of security group IDs.
Core and Task Security Group	Optional. ID of the security group for the cluster core and task nodes. When the AWS administrator creates and configures a cluster In the AWS console, the property is equivalent to the ElasticMapReduce-slave security group If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Core and Task Security Groups	Optional. IDs of additional security groups to attach to cluster core and task nodes. Use a comma-separated list of security group IDs.
Service Access Security Group	EMR managed security group for service access. Required when you provision an EMR cluster in a private subnet.

Azure Cloud Provisioning Configuration Properties

The properties in the Azure cloud provisioning configuration enable the Data Integration Service to contact and create resources on the Azure cloud platform.

Authentication Details

The following table describes authentication properties to configure:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.

Property	Description
Description	Optional. Description of the cloud provisioning configuration.
Subscription ID	ID of the Azure account to use in the cluster creation process.
Tenant ID	A GUID string associated with the Azure Active Directory.
Client ID	A GUID string that is the same as the Application ID associated with the Service Principal. The Service Principal must be assigned to a role that has permission to create resources in the subscription that you identified in the Subscription ID property.
Client Secret	An octet string that provides a key associated with the client ID.

Storage Account Details

Choose to configure access to one of the following storage types:

- Azure Data Lake Storage (ADLS). See Azure documentation.
- An Azure Storage Account, known as general or blob storage. See Azure documentation.

The following table describes the information you need to configure Azure Data Lake Storage (ADLS) with the HDInsight cluster:

Property	Description
Azure Data Lake Store Name	Name of the ADLS storage to access. The ADLS storage and the cluster to create must reside in the same region.
Data Lake Service Principal Client ID	A credential that enables programmatic access to ADLS storage. Enables the Informatica domain to communicate with ADLS and run commands and mappings on the HDInsight cluster. The service principal is an Azure user that meets the following requirements: - Permissions to access required directories in ADLS storage. - Certificate-based authentication for ADLS storage. - Key-based authentication for ADLS storage.
Data Lake Service Principal Certificate Contents	The Base64 encoded text of the public certificate used with the service principal. Leave this property blank when you create the cloud provisioning configuration. After you save the cloud provisioning configuration, log in to the VM where the Informatica domain is installed and run infacmd ccps updateADLSCertificate to populate this property.
Data Lake Service Principal Certificate Password	Private key for the service principal. This private key must be associated with the service principal certificate.
Data Lake Service Principal Client Secret	An octet string that provides a key associated with the service principal.
Data Lake Service Principal OAUTH Token Endpoint	Endpoint for OAUTH token based authentication.

The following table describes the information you need to configure Azure General Storage, also known as blob storage, with the HDInsight cluster:

Property	Description
Azure Storage Account Name	Name of the storage account to access. Get the value from the Storage Accounts node in the Azure web console. The storage and the cluster to create must reside in the same region.
Azure Storage Account Key	A key to authenticate access to the storage account. To get the value from the Azure web console, select the storage account, then Access Keys. The console displays the account keys.

Cluster Deployment Details

The following table describes the cluster deployment properties that you configure:

Property	Description
Resource Group	Resource group in which to create the cluster. A resource group is a logical set of Azure resources.
Virtual Network Resource Group	Optional. Resource group to which the virtual network belongs. If you do not specify a resource group, the Data Integration Service assumes that the virtual network is a member of the same resource group as the cluster.
Virtual Network	Name of the virtual network or vnet where you want to create the cluster. Specify a vnet that resides in the resource group that you specified in the Virtual Network Resource Group property. The vnet must be in the same region as the region in which to create the cluster.
Subnet Name	Subnet in which to create the cluster. The subnet must be a part of the vnet that you designated in the previous property. Each vnet can have one or more subnets. The Azure administrator can choose an existing subnet or create one for the cluster.

External Hive Metastore Details

You can specify the properties to enable the cluster to connect to a Hive metastore database that is external to the cluster.

If you do not specify an existing external database in this dialog box, the cluster creates its own database on the cluster. This database is terminated when the cluster is terminated.

You can use an external relational database like MySQL or Amazon RDS as the Hive metastore database. The external database must be on the same cloud platform as the cluster to create.

The following table describes the Hive metastore database properties that you configure:

Property	Description
Database Name	Name of the Hive metastore database.
Database Server Name	Server on which the database resides. **Note:** The database server name on the Azure web console commonly includes the suffix `database.windows.net`. For example: `server123xyz.database.windows.net`. You can specify the database server name without the suffix and Informatica will automatically append the suffix. For example, you can specify `server123xyz`.

Property	Description
Database User Name	User name of the account for the domain to use to access the database.
Database Password	Password for the user account.

Authentication Details

The following table describes authentication properties to configure:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description	Optional. Description of the cloud provisioning configuration.
Subscription ID	ID of the Azure account to use in the cluster creation process.
Tenant ID	A GUID string associated with the Azure Active Directory.
Client ID	A GUID string that is the same as the Application ID associated with the Service Principal. The Service Principal must be assigned to a role that has permission to create resources in the subscription that you identified in the Subscription ID property.
Client Secret	An octet string that provides a key associated with the client ID.

Storage Account Details

Choose to configure access to one of the following storage types:

- Azure Data Lake Storage (ADLS). See Azure documentation.
- An Azure Storage Account, known as general or blob storage. See Azure documentation.

The following table describes the information you need to configure Azure Data Lake Storage (ADLS) with the HDInsight cluster:

Property	Description
Azure Data Lake Store Name	Name of the ADLS storage to access. The ADLS storage and the cluster to create must reside in the same region.
Data Lake Service Principal Client ID	A credential that enables programmatic access to ADLS storage. Enables the Informatica domain to communicate with ADLS and run commands and mappings on the HDInsight cluster. The service principal is an Azure user that meets the following requirements: - Permissions to access required directories in ADLS storage. - Certificate-based authentication for ADLS storage. - Key-based authentication for ADLS storage.
Data Lake Service Principal Certificate Contents	The Base64 encoded text of the public certificate used with the service principal. Leave this property blank when you create the cloud provisioning configuration. After you save the cloud provisioning configuration, log in to the VM where the Informatica domain is installed and run infacmd ccps updateADLSCertificate to populate this property.
Data Lake Service Principal Certificate Password	Private key for the service principal. This private key must be associated with the service principal certificate.
Data Lake Service Principal Client Secret	An octet string that provides a key associated with the service principal.
Data Lake Service Principal OAUTH Token Endpoint	Endpoint for OAUTH token based authentication.

The following table describes the information you need to configure Azure General Storage, also known as blob storage, with the HDInsight cluster:

Property	Description
Azure Storage Account Name	Name of the storage account to access. Get the value from the Storage Accounts node in the Azure web console. The storage and the cluster to create must reside in the same region.
Azure Storage Account Key	A key to authenticate access to the storage account. To get the value from the Azure web console, select the storage account, then Access Keys. The console displays the account keys.

Cluster Deployment Details

The following table describes the cluster deployment properties that you configure:

Property	Description
Resource Group	Resource group in which to create the cluster. A resource group is a logical set of Azure resources.
Virtual Network Resource Group	Optional. Resource group to which the virtual network belongs. If you do not specify a resource group, the Data Integration Service assumes that the virtual network is a member of the same resource group as the cluster.
Virtual Network	Name of the virtual network or vnet where you want to create the cluster. Specify a vnet that resides in the resource group that you specified in the Virtual Network Resource Group property. The vnet must be in the same region as the region in which to create the cluster.
Subnet Name	Subnet in which to create the cluster. The subnet must be a part of the vnet that you designated in the previous property. Each vnet can have one or more subnets. The Azure administrator can choose an existing subnet or create one for the cluster.

External Hive Metastore Details

You can specify the properties to enable the cluster to connect to a Hive metastore database that is external to the cluster.

You can use an external relational database like MySQL or Amazon RDS as the Hive metastore database. The external database must be on the same cloud platform as the cluster to create.

If you do not specify an existing external database in this dialog box, the cluster creates its own database on the cluster. This database is terminated when the cluster is terminated.

The following table describes the Hive metastore database properties that you configure:

Property	Description
Database Name	Name of the Hive metastore database.
Database Server Name	Server on which the database resides. **Note:** The database server name on the Azure web console commonly includes the suffix `database.windows.net`. For example: `server123xyz.database.windows.net`. You can specify the database server name without the suffix and Informatica will automatically append the suffix. For example, you can specify `server123xyz`.
Database User Name	User name of the account for the domain to use to access the database.
Database Password	Password for the user account.

Databricks Cloud Provisioning Configuration Properties

The properties in the Databricks cloud provisioning configuration enable the Data Integration Service to contact and create resources on the Databricks cloud platform.

The following table describes the Databricks cloud provisioning configuration properties:

Property	Description
Name	Name of the cloud provisioning configuration. **Tip:** Because the Administrator tool lists cloud provisioning configuration objects with other connections, use a naming convention such as "CPC" as part of the name of the object to help identify it.
ID	The cluster ID of the Databricks cluster.
Description	Optional description of the cloud provisioning configuration.
Databricks domain	Domain name of the Databricks deployment.
Databricks token ID	The token ID created within Databricks 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.
Advanced Properties	Advanced properties that are unique to the Databricks cloud provisioning configuration.

Advanced Properties

Configure the following properties in the **Advanced Properties** of the Databricks configuration section:

infaspark.pythontx.exec

Required to run a Python transformation on the Databricks Spark engine. Set to the location of the Python executable binary on the worker nodes in the Databricks cluster.

When you provision the cluster at run time, set this property in the Databricks cloud provisioning configuration. Otherwise, set on the Databricks connection.

For example, set to:

```
infaspark.pythontx.exec=/databricks/python3/bin/python3
```

infaspark.pythontx.executorEnv.PYTHONHOME

Required to run a Python transformation on the Databricks Spark engine. Set to the location of the Python installation directory on the worker nodes in the Databricks cluster.

When you provision the cluster at run time, set this property in the Databricks cloud provisioning configuration. Otherwise, set on the Databricks connection.

For example, set to:

```
infaspark.pythontx.executorEnv.PYTHONHOME=/databricks/python3
```

Create the Cloud Provisioning Configuration

Create the cloud provisioning configuration and configure it with information that the domain needs to access and create resources on the cloud platform.

The properties to configure in the cloud provisioning configuration depend on the cloud platform.

1. From the **Connections** tab, right-click the Domain node and select **New** > **Connection**.

   The **New Connection** dialog box opens.
2. Choose one of the following cloud provisioning configuration types:
   - AWS Cloud Provisioning Configuration. For AWS cloud provisioning properties and values, see "AWS Cloud Provisioning Configuration Properties" on page 65.
   - Azure Cloud Provisioning Configuration. For Azure cloud provisioning properties and values, see "Azure Cloud Provisioning Configuration Properties" on page 68.
   - Databricks Cloud Provisioning Configuration. For Databricks cloud provisioning properties and values, see "Databricks Cloud Provisioning Configuration Properties" on page 74
3. Enter property values in the configuration wizard, then click **Finish** to create the cloud provisioning configuration.

The cloud provisioning configuration appears in the list of connections in the **Domain Navigator**.

Complete the Azure Cloud Provisioning Configuration

When you want to access Azure Data Lake Storage (ADLS) with the cluster workflow, complete the following steps after you configure and save the cloud provisioning configuration for Azure:

1. Log in to the VM where the Informatica domain is installed, and open a command shell.
2. From the command line, issue the following command:

   ```
   /infacmd.sh ccps updateADLSCertificate -dn <domain name> -un <user name> -pd
   <password> -cpcid <cloud provisioning connection name>
   -certPath <domain location of certificate>
   ```

   The command automatically populates the Data Lake Service Principal Certificate Contents property of the cloud provisioning connection.

Create a Cluster Connection

Create a dedicated cluster connection, such as Hadoop or Databricks, to use with the cluster workflow.

The cluster connection saves property values for the Data Integration Service to use for cluster workflow operations. When you run a cluster workflow, the Data Integration Service uses settings in the cluster connection to run jobs on the cluster.

When you configure the connection, populate the Cloud Provisioning Configuration property with the name of the cloud provisioning configuration that you created for the workflow. Leave the Cluster Configuration property blank.

When you create the workflow, populate the Connection Name property of the Create Cluster task with this cluster connection.

For information about creating connections in the Administrator tool, see the Connections Reference in the *Data Engineering Administrator Guide*.

CHAPTER 7

Data Integration Service Processing

This chapter includes the following topics:

Overview of Data Integration Service Processing

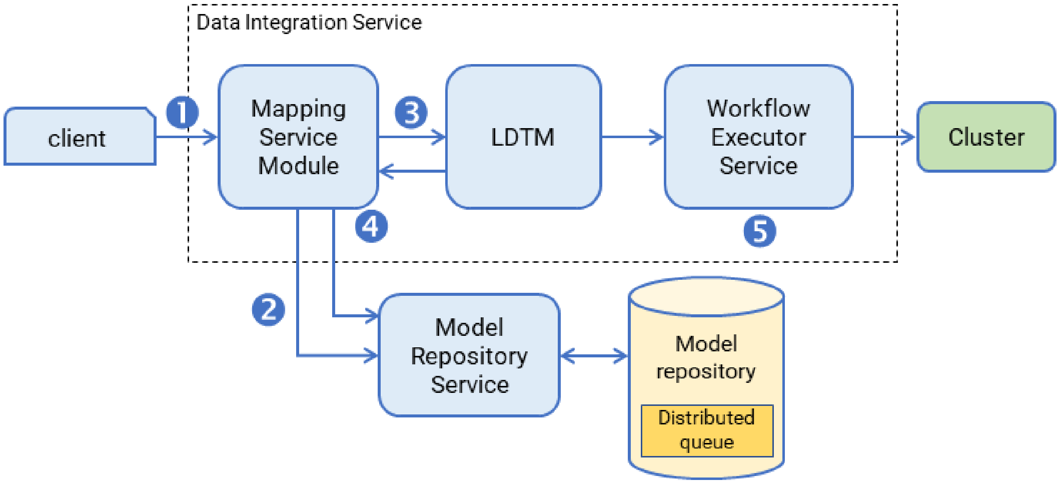
The Data Integration Service runs mappings, workflows, and other jobs that a developer deploys from clients. The Data Integration Service uses an internal process to assess the job, create an execution plan, and process the job as a series of tasks. Depending on how you configure the Data Integration Service, the process runs the tasks on the Data Integration Service or sends the job for processing to a compute cluster.

When you run a mapping or other job on the Data Integration Service, the Data Integration Service saves the mapping request in a queue. The Data Integration Service mapping service takes the job from the queue when nodes are available to run the job, and creates an execution workflow. The Data Integration Service then processes the execution workflow natively or sends workflow tasks to the cluster for processing.

When a compute cluster receives a job from the Data Integration Service, it assigns each job and each child task a YARN ID. The Data Integration Service stores the YARN ID in the Model repository database to aid in tracking jobs.

The job is complete when the client receives the Complete status from the Data Integration Service.

The following image shows an overview of Data Integration Service processing:



1. A client submits a mapping execution request to the Data Integration Service. The Mapping Service Module receives the request and stores the job in the queue.
2. The Mapping Service Module connects to the Model Repository Service to fetch mapping metadata from the Model repository.
3. The Mapping Service Module passes the mapping to the Logical Data Transformation Manager (LDTM).
4. The LDTM compiles the mapping and generates the Spark execution workflow. It stores the execution workflow in the Model repository.
5. The LTDM pushes the execution workflow through the Workflow Executor Service to the cluster for processing.

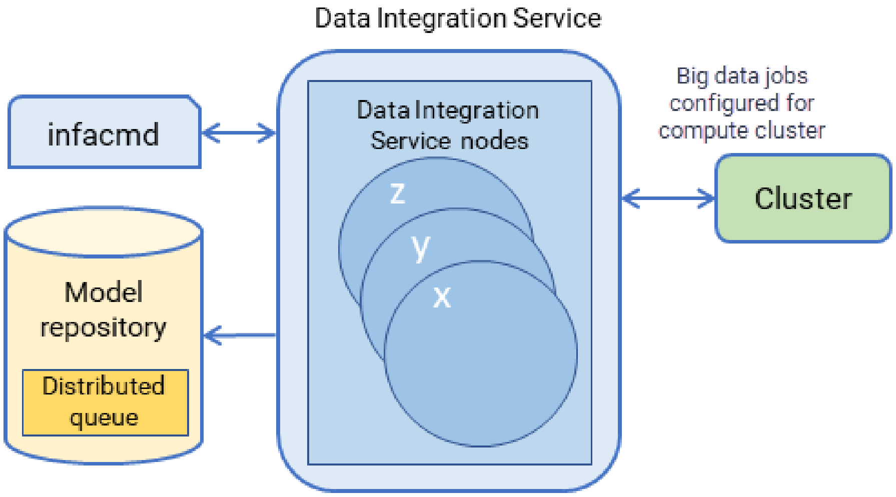
You can tune the Data Integration Service and run-time engines for large dataset processing to ensure that sufficient resources are available to perform jobs.

Data Integration Service Queueing

The Data Integration Service uses a distributed queue to store job information until resources are available to run the job. The distributed queue is stored in the Model repository and is shared by the backup node, if one exists, or by all nodes in the grid.

When you run a mapping job or workflow mapping task, the Data Integration Service adds the job to the queue. The job state appears as "Queued" in the Administrator tool contents panel. When resources are available, the Data Integration Service takes a job from the queue and runs it.

The following image shows the location of the distributed queue:


Data Integration Service
infacmd
Data Integration Service nodes
z
y
x
Big data jobs configured for compute cluster
Cluster
Model repository
Distributed queue

Consider the following queueing process:

1. A client submits a job request to the Data Integration Service, which stores job metadata in the distributed queue.
2. When the Data Integration Service node has available resources, the Data Integration Service retrieves the job from the queue and sends it to the available node for processing.
3. If a node fails while running a job, the job can fail over to another node. Any back-up node or node in the grid can take jobs from the queue.
4. The interrupted job runs on the new node.

When you run a job that cannot be queued, the Data Integration Service immediately starts running the job. If there are not enough resources available, the job fails, and you must run the job again when resources are available.

The following jobs cannot be queued:

- Jobs that cannot be deployed, such as previews and profiles
- On-demand jobs
- SQL queries
- Web service requests

You can use the command infacmd ms abortAllJobs to abort all jobs in the queue, or infacmd ms purgeDatabaseWorkTables to clear the queue.

Execution Pools

Execution pools define the number of jobs that can run concurrently on the Data Integration Service.

Configure execution pool sizes to limit the number of jobs that can run concurrently. You can configure native, Hadoop, and on-demand execution pool sizes.

The execution pool size acts as a threshold for concurrent jobs. When the number of job requests exceeds the threshold, the Data Integration Service stops accepting new jobs.

When the number of concurrent jobs exceeds the native and Hadoop execution pool size, the Data Integration Service stops taking jobs from the queue until the number of concurrently running jobs falls below the threshold you configured.

When the number of concurrent jobs exceeds the on-demand execution pool threshold, the Data Integration Service rejects on-demand job requests. When the number of concurrently running jobs falls below the threshold you configured, you can resume running on-demand jobs and re-run rejected jobs.

Configure execution pool sizes in Data Integration Service Execution Options settings.

Data Engineering Recovery

The Data Integration Service manages jobs that are deployed to run in a cluster environment. When you enable the Data Integration Service for data engineering recovery, the Data Integration Service can recover and continue processing jobs that run on the Spark engine.

To use data engineering recovery, you must configure jobs to run on the Spark engine. Configure the Data Integration Service and log settings, and run the job from infacmd.

The Data Integration Service maintains a queue of jobs to run. The Data Integration Serviced assigns jobs from the queue to nodes which prepare them and send them to a compute cluster for processing.

The cluster assigns a YARN ID to each job and each of its child tasks to track jobs as it runs them. The Data Integration Service gets the YARN IDs from the cluster and stores them on the Model repository database.

If the Data Integration Service runs on a grid or multiple nodes, when a node fails, the Service Manager fails over to another node. The Data Integration Service queries the cluster for the status of tasks as identified by their YARN IDs and compares the response with the status of failed over tasks. Depending on the status, the Data Integration Service takes the following actions:

- If a task has no YARN ID, it submits the task to the cluster.
- If a task that has a YARN ID has not been sent to the cluster, it submits the task for processing.
- If all tasks have been sent, it continues to monitor communications from the cluster until completion.

If the Data Integration Service runs on a single node, it attempts job recovery when the node is restored.

When the Data Integration Service restarts and runs a job, the job creates a cluster configuration under the `disTemp` directory. This process causes the `disTemp` directory to grow over time. Manage disk space by monitoring and periodically clearing the contents of the `disTemp` directory.

Note: The Data Integration Service begins the recovery process by verifying that inactive nodes are not available, and then it assigns the recovered job to an available node. The verification process for unavailable nodes might take several minutes before the job is reassigned to an available node.

Scenarios Where Recovery is Possible

The Data Integration Service can recover queued or running jobs upon node failure in the following scenarios:

The job has been taken from the queue but is not yet submitted to the cluster.

Depending on the timing of node failure, the status might be Queued or Running. The Data Integration Service fails the job over to the restored node or another node and continues processing it from the point of interruption.

The job is partially submitted to the cluster.

The status of tasks sent to the cluster is Running, and the status of tasks in the queue remains Queued. The Data Integration Service identifies unsent tasks and sends them to the restored node or another node, which processes the unsent tasks and sends them to the cluster for processing.

The job is fully submitted to the cluster.

The Data Integration Service has completed its job processing, and node failure at this point has no effect on the queue or on Hadoop processing. The Data Integration Service fails over to a restored node or another node to continue communication with Hadoop and to monitor job status.

Note: If the Data Integration Service runs on a single node, the recovery process begins when the node is running again.

Scenarios Where Recovery is Not Possible

Job recovery is not possible in the following scenarios:

- The node shuts down before the Mapping Service Module stores the job request in the Model repository.
  The job status depends on the timing of the failure. Either the job does not appear at all, or the status is Unknown.
- You cancel a mapping while it is running.
  The job status depends on the timing of the cancellation request. The status will be Canceled or Complete.
- You recycle the Data Integration Service while a job is running.
  The job status will be Aborted or Canceled.
- The cluster fails, or communication with the cluster is lost while a job is running.
  The job status will be Failed.

In each case, the job must be manually re-run.

When a job has a status of Unknown, check the Data Integration Service log and check for the existence of a session log for more information.

Note: Depending on the size of the loan on the Data Integration Service grid, logs might not be updated with failed job states for several minutes.

Recovery Job Management

When you manually interrupt the Data Integration Service, the recovery jobs might not be aborted, and they might not be removed from the Model repository queue. Manual interruption includes disabling, recycling, or deleting the service. You can manage jobs associated with manual interruption through infacmd.

Consider the following ways to manage recovery jobs when you recycle, disable, or delete the Data Integration Service:

Abort jobs in the queue

If you recycle or disable the Data Integration Service on a grid while jobs are running, the service might attempt recovery when it is enabled.

If you do not want to recover the jobs, run infacmd ms abortAllJobs before you recycle or disable the service.

Note: If you recycle or disable a Data Integration Service on a single node or on primary and backup nodes, the service aborts all jobs and does not attempt recovery when it is enabled.

Purge jobs in the queue

When you recycle or delete a Data Integration Service, the queue in the Model repository retains job information stored for recovery.

To maintain the queue and purge the jobs, run infacmd ms PurgeDatabaseWorktables.

For more information about the commands, see the *Informatica Command Reference*.

Monitoring Recovered Jobs

You can identify recovered jobs on the Monitoring tab.

When a job is recovered, the Monitoring tab shows the same start and end time for the job, and the elapsed time = 0. While this statistic is not the actual elapsed time, it enables you to identify jobs that were recovered. For a more accurate view of the elapsed time for the job, view the Spark job logs on the cluster or the session logs on the Data Integration Service.

Consider the following behavior:

- When the Data Integration Service recovers a job, the Administrator tool might display incomplete job statistics in the Monitoring tab when the job is complete. For example, the job statistics might not correctly display the number of rows processed.
- The Monitoring tab does not display detailed statistics if the Data Integration Service process stops after it submits a job to the compute cluster.

Data Engineering Recovery Configuration

Configure a Data Integration Service for data engineering recovery. If the Data Integration Service runs on a grid, verify that certain directories are configured on a network file system.

Perform the following tasks:

Enable data engineering recovery.

For the Data Integration Service to recover mapping jobs, you must enable the option in the Data Integration Service Execution Options.

Verify that directories are configured on a network file system.

If the Data Integration Service runs on a grid, some directories on each node must be configured on a network file system to permit the Data Integration Service components to communicate information about mapping jobs during node failures.

Verify that the following properties are configured for directories:

- Directory for temporary files. Default is `<home directory/disTemp>`
- Log directory. Default is `<Informatica installation directory/logs/<node name>/services/DataIntegrationService`

Note: Directories must be mounted on a network file system accessible to all Data Integration Service nodes.

Tuning for Data Engineering Job Processing

Tune the application services and run-time engines for processing data engineering jobs.

You might want to tune the application services and run-time engines for data engineering job processing to ensure that the application services and the run-time engines are allocated enough resources to perform jobs.

For example, the Model Repository Service and the Data Integration Service require resources to store run-time data. When you run mappings, you might deploy the mappings to the Data Integration Service and run the mappings on the Blaze engine. Similarly, the Blaze engine requires resources to run the mappings. You must allocate enough resources between the Model Repository Service, the Data Integration Service, and the Blaze engine to ensure that mapping performance is optimized.

You can tune the application services and run-time engines based on deployment type. A deployment type represents job processing requirements based on concurrency and volume. The deployment type defines the amount of resources that application services and run-time engines require to function efficiently, and how resources should be allocated between the application services and run-time engines.

To tune the application services and run-time engines, assess the deployment type that best describes the environment that you use for processing data engineering jobs. Then select the application services and the run-time engines that you want to tune. Tune the application services and the run-time engines using `infacmd autotune autotune`.

Deployment Types

A deployment type represents big data processing requirements based on concurrency and volume.

The deployment type defines the amount of resources that application services and run-time engines require to function efficiently, and how resources should be allocated between the application services and run-time engines. The deployment types are Sandbox, Basic, Standard, and Advanced.

The following table describes the deployment types:

Deployment Type	Description
Sandbox	Used for proof of concepts or as a sandbox with minimal users.
Basic	Used for low volume processing with low levels of concurrency.
Standard	Used for high volume processing with low levels of concurrency.
Advanced	Used for high volume processing with high levels of concurrency.

Each deployment type is described using deployment criteria. The deployment criteria is a set of characteristics that are common to each deployment type. Use the deployment criteria to help you understand the deployment type that best fits the environment that you use for big data processing.

The following table defines the deployment criteria:

Deployment Criteria	Sandbox	Basic	Standard	Advanced
Total data volume	2-10 GB	10 GB - 2 TB	2 TB - 50 TB	50 TB+
Number of nodes in the Hadoop environment	2 - 5	5 - 25	25 - 100	100+
Number of developers	2	5	10	50
Number of concurrent jobs in the Hadoop environment	< 10	< 250	< 500	2000+
Number of Model repository objects	<1000	< 5000	5000 - 20000	20000+
Number of deployed applications	< 10	< 25	< 100	< 500
Number of objects per deployed application	< 10	< 50	< 100	< 100

For example, you estimate that your environment handles an average of 400 concurrent jobs and a data volume of 35 TB. According to the deployment criteria, the deployment type that best describes your environment is Standard.

Tuning the Application Services

Tune the application services for big data processing.

You tune the application services according to the deployment type that best describes the big data processing requirements in your environment. For each application service, the heap memory is tuned based on the deployment type.

The following table describes how the heap memory is tuned for each application service based on the deployment type:

Service	Sandbox	Basic	Standard	Advanced
Analyst Service	768 MB	1 GB	2 GB	4 GB
Content Management Service	1 GB	2 GB	4 GB	4 GB
Data Integration Service	640 MB	2 GB	4 GB	6 GB
Model Repository Service	1 GB	1 GB	2 GB	4 GB
Resource Manager Service	512 MB	512 MB	2 GB	4 GB
Search Service	768 MB	1 GB	2 GB	4 GB

Data Integration Service

When you tune the Data Integration Service, the deployment type additionally defines the execution pool size for jobs that run in the native and Hadoop environments.

The following table lists the execution pool size that is tuned in the native and Hadoop environments based on the deployment type:

Run-time Environment	Sandbox	Basic	Standard	Advanced
Native	10	10	15	30
Hadoop	10	500	1000	2000

Note: If the deployment type is Advanced, the Data Integration Service is tuned to run on a grid.

Tuning the Hadoop Run-time Engines

Tune the Blaze and Spark engines based on the deployment type. You tune the Blaze and Spark engines to adhere to big data processing requirements.

Tuning the Spark Engine

Tune the Spark engine according to a deployment type that defines the big data processing requirements on the Spark engine. When you tune the Spark engine, the autotune command configures the Spark advanced properties in the Hadoop connection.

The following table describes the advanced properties that are tuned:

Property	Description
spark.driver.memory	The driver process memory that the Spark engine uses to run mapping jobs.
spark.executor.memory	The amount of memory that each executor process uses to run tasklets on the Spark engine.
spark.executor.cores	The number of cores that each executor process uses to run tasklets on the Spark engine.
spark.sql.shuffle.partitions	The number of partitions that the Spark engine uses to shuffle data to process joins or aggregations in a mapping job.

The following table lists the tuned value for each advanced property based on the deployment type:

Property	Sandbox	Basic	Standard	Advanced
spark.driver.memory	1 GB	2 GB	4 GB	4 GB
spark.executor.memory	2 GB	4 GB	6 GB	6 GB
spark.executor.cores	2	2	2	2
spark.sql.shuffle.partitions	100	400	1500	3000

Tuning the Blaze Engine

Tune the Blaze engine to adhere to big data processing requirements on the Blaze engine. When you tune the Blaze engine, the autotune command configures the Blaze advanced properties in the Hadoop connection.

The following table describes the Blaze properties that are tuned:

Property	Description	Value
infagrid.orch.scheduler.oop.container.pref.memory	The amount of memory that the Blaze engine uses to run tasklets.	5120
infagrid.orch.scheduler.oop.container.pref.vcore	The number of DTM instances that run on the Blaze engine.	4
infagrid.tasklet.dtm.buffer.block.size	The amount of buffer memory that a DTM instance uses to move a block of data in a tasklet.	6553600
** The tuned properties do not depend on the deployment type.*		

Autotune

Configures services and connections with recommended settings based on the deployment type. Changes take effect after you recycle the services.

For each specified service, the changes to the service take effect on all nodes that are currently configured to run the service, and the changes affect all service processes.

The infacmd autotune Autotune command uses the following syntax:

```
Autotune

<-DomainName|-dn> domain_name
<-UserName|-un> user_name
<-Password|-pd> password
[<-SecurityDomain|-sdn> security_domain]
[<-ResilienceTimeout|-re> timeout_period_in_seconds]
<-Size|-s> tuning_size_name
[<-ServiceNames|-sn> service_names]
[<-BlazeConnectionNames|-bcn> connection_names]
[<-SparkConnectionNames|-scn> connection_names]
[<-All|-a> yes_or_no]
```

The infacmd program uses the following common options to connect to the domain: domain name, user name, password, security domain, and resilience timeout. The table of options has brief descriptions. To see more information about connecting to the domain, see the Command Reference.

The following table describes infacmd autotune Autotune options and arguments:

Option	Description
-DomainName -dn	Name of the Informatica domain.
-UserName -un	User name to connect to the domain.
-Password -pd	Password for the user name.

Option	Description
-SecurityDomain -sdn	Name of the security domain to which the domain user belongs.
-ResilienceTimeout -re	Amount of time in seconds that infacmd attempts to establish or re-establish a connection to the domain.
-Size -s	Required. The deployment type that represents big data processing requirements based on concurrency and volume. You can enter Sandbox, Basic, Standard, or Advanced.
-ServiceNames -sn	Optional. List of services configured in the Informatica domain. Separate each service name with a comma. You can tune the following services: - Analyst Service - Content Management Service - Data Integration Service - Model Repository Service - Resource Manager Service - Search Service Default is none.
-BlazeConnectionNames -bcn	Optional. List of Hadoop connections configured in the Informatica domain. For each Hadoop connection, the command tunes Blaze configuration properties in the Hadoop connection. Separate each Hadoop connection name with a comma. Default is none.
-SparkConnectionNames -scn	Optional. List of Hadoop connections configured in the Informatica domain. For each Hadoop connection, the command tunes Spark configuration properties in the Hadoop connection. Separate each Hadoop connection name with a comma. Default is none.
-All -a	Optional. Enter `yes` to apply recommended settings to all Analyst Services, Content Management Services, Data Integration Services, Model Repository Services, Resource Manager Services, Search Services, and Hadoop connections in the Informatica domain. Enter `no` to apply the recommended settings only to the services and Hadoop connections that you specify. Default is `no`.

APPENDIX A

Connections Reference

This appendix includes the following topics:

Connections Overview

Create a connection to access non-native environments, Hadoop and Databricks. If you access HBase, HDFS, or Hive sources or targets in the Hadoop environment, you must also create those connections. You can create the connections using the Developer tool, Administrator tool, and infacmd.

You can create the following types of connections:

Hadoop connection

Create a Hadoop connection to run mappings in the Hadoop environment.

HBase connection

Create an HBase connection to access HBase. The HBase connection is a NoSQL connection.

HDFS connection

Create an HDFS connection to read data from or write data to the HDFS file system on a Hadoop cluster.

Hive connection

Create a Hive connection to access Hive as a source or target. You can access Hive as a source if the mapping is enabled for the native or Hadoop environment. You can access Hive as a target if the mapping runs on the Blaze engine.

JDBC connection

Create a JDBC connection and configure Sqoop properties in the connection to import and export relational data through Sqoop.

Databricks connection

Create a Databricks connection to run mappings in the Databricks environment.

Note: For information about creating connections to other sources or targets such as social media web sites or Teradata, see the respective PowerExchange adapter user guide for information.

Cloud Provisioning Configuration

The cloud provisioning configuration establishes a relationship between the Create Cluster task and the cluster connection that the workflows use to run mapping tasks. The Create Cluster task must include a reference to the cloud provisioning configuration. In turn, the cloud provisioning configuration points to the cluster connection that you create for use by the cluster workflow.

The properties to populate depend on the Hadoop distribution you choose to build a cluster on. Choose one of the following connection types:

- AWS Cloud Provisioning. Connects to an Amazon EMR cluster on Amazon Web Services.
- Azure Cloud Provisioning. Connects to an HDInsight cluster on the Azure platform.
- Databricks Cloud Provisioning. Connects to a Databricks cluster on the Azure Databricks platform.

AWS Cloud Provisioning Configuration Properties

The properties in the AWS cloud provisioning configuration enable the Data Integration Service to contact and create resources on the AWS cloud platform.

General Properties

The following table describes cloud provisioning configuration general properties:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description.	Optional. Description of the cloud provisioning configuration.
AWS Access Key ID	Optional. ID of the AWS access key, which AWS uses to control REST or HTTP query protocol requests to AWS service APIs. If you do not specify a value, Informatica attempts to follow the Default Credential Provider Chain.
AWS Secret Access Key	Secret component of the AWS access key. Required if you specify the AWS Access Key ID.
Region	Region in which to create the cluster. This must be the region in which the VPC is running. Use AWS region values. For a list of acceptable values, see AWS documentation. **Note:** The region where you want to create the cluster can be different from the region in which the Informatica domain is installed.

Permissions

The following table describes cloud provisioning configuration permissions properties:

Property	Description
EMR Role	Name of the service role for the EMR cluster that you create. The role must have sufficient permissions to create a cluster, access S3 resources, and run jobs on the cluster. When the AWS administrator creates this role, they select the "EMR" role. This contains the default AmazonElasticMapReduceRole policy. You can edit the services in this policy.
EC2 Instance Profile	Name of the EC2 instance profile role that controls permissions on processes that run on the cluster. When the AWS administrator creates this role, they select the "EMR Role for EC2" role. This includes S3 access by default.
Auto Scaling Role	Required if you configure auto-scaling for the EMR cluster. This role is created when the AWS administrator configures auto-scaling on any cluster in the VPC. Default: When you leave this field blank, it is equivalent to setting the Auto Scaling role to "Proceed without role" when the AWS administrator creates a cluster in the AWS console.

EC2 Configuration

The following table describes cloud provisioning configuration EC2 configuration properties:

Property	Description
EC2 Key Pair	EC2 key pair to enable communication with the EMR cluster master node. Optional. This credential enables you to log into the cluster. Configure this property if you intend the cluster to be non-ephemeral.
EC2 Subnet	ID of the subnet on the VPC in which to create the cluster. Use the subnet ID of the EC2 instance where the cluster runs.
Master Security Group	Optional. ID of the security group for the cluster master node. Acts as a virtual firewall to control inbound and outbound traffic to cluster nodes. Security groups are created when the AWS administrator creates and configures a cluster in a VPC. In the AWS console, the property is equivalent to ElasticMapReduce-master. You can use existing security groups, or the AWS administrator might create dedicated security groups for the ephemeral cluster. If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Master Security Groups	Optional. IDs of additional security groups to attach to the cluster master node. Use a comma-separated list of security group IDs.
Core and Task Security Group	Optional. ID of the security group for the cluster core and task nodes. When the AWS administrator creates and configures a cluster In the AWS console, the property is equivalent to the ElasticMapReduce-slave security group If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Core and Task Security Groups	Optional. IDs of additional security groups to attach to cluster core and task nodes. Use a comma-separated list of security group IDs.
Service Access Security Group	EMR managed security group for service access. Required when you provision an EMR cluster in a private subnet.

General Properties

The following table describes cloud provisioning configuration general properties:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description.	Optional. Description of the cloud provisioning configuration.
AWS Access Key ID	Optional. ID of the AWS access key, which AWS uses to control REST or HTTP query protocol requests to AWS service APIs. If you do not specify a value, Informatica attempts to follow the Default Credential Provider Chain.

Property	Description
AWS Secret Access Key	Secret component of the AWS access key. Required if you specify the AWS Access Key ID.
Region	Region in which to create the cluster. This must be the region in which the VPC is running. Use AWS region values. For a list of acceptable values, see AWS documentation. **Note:** The region where you want to create the cluster can be different from the region in which the Informatica domain is installed.

Permissions

The following table describes cloud provisioning configuration permissions properties:

Property	Description
EMR Role	Name of the service role for the EMR cluster that you create. The role must have sufficient permissions to create a cluster, access S3 resources, and run jobs on the cluster. When the AWS administrator creates this role, they select the “EMR” role. This contains the default AmazonElasticMapReduceRole policy. You can edit the services in this policy.
EC2 Instance Profile	Name of the EC2 instance profile role that controls permissions on processes that run on the cluster. When the AWS administrator creates this role, they select the “EMR Role for EC2” role. This includes S3 access by default.
Auto Scaling Role	Required if you configure auto-scaling for the EMR cluster. This role is created when the AWS administrator configures auto-scaling on any cluster in the VPC. Default: When you leave this field blank, it is equivalent to setting the Auto Scaling role to “Proceed without role” when the AWS administrator creates a cluster in the AWS console.

EC2 Configuration

The following table describes cloud provisioning configuration EC2 configuration properties:

Property	Description
EC2 Key Pair	EC2 key pair to enable communication with the EMR cluster master node. Optional. This credential enables you to log into the cluster. Configure this property if you intend the cluster to be non-ephemeral.
EC2 Subnet	ID of the subnet on the VPC in which to create the cluster. Use the subnet ID of the EC2 instance where the cluster runs.
Master Security Group	Optional. ID of the security group for the cluster master node. Acts as a virtual firewall to control inbound and outbound traffic to cluster nodes. Security groups are created when the AWS administrator creates and configures a cluster in a VPC. In the AWS console, the property is equivalent to ElasticMapReduce-master. You can use existing security groups, or the AWS administrator might create dedicated security groups for the ephemeral cluster. If you do not specify a value, the cluster applies the default security group for the VPC.

Property	Description
Additional Master Security Groups	Optional. IDs of additional security groups to attach to the cluster master node. Use a comma-separated list of security group IDs.
Core and Task Security Group	Optional. ID of the security group for the cluster core and task nodes. When the AWS administrator creates and configures a cluster In the AWS console, the property is equivalent to the ElasticMapReduce-slave security group If you do not specify a value, the cluster applies the default security group for the VPC.
Additional Core and Task Security Groups	Optional. IDs of additional security groups to attach to cluster core and task nodes. Use a comma-separated list of security group IDs.
Service Access Security Group	EMR managed security group for service access. Required when you provision an EMR cluster in a private subnet.

Azure Cloud Provisioning Configuration Properties

The properties in the Azure cloud provisioning configuration enable the Data Integration Service to contact and create resources on the Azure cloud platform.

Authentication Details

The following table describes authentication properties to configure:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description	Optional. Description of the cloud provisioning configuration.
Subscription ID	ID of the Azure account to use in the cluster creation process.
Tenant ID	A GUID string associated with the Azure Active Directory.
Client ID	A GUID string that is the same as the Application ID associated with the Service Principal. The Service Principal must be assigned to a role that has permission to create resources in the subscription that you identified in the Subscription ID property.
Client Secret	An octet string that provides a key associated with the client ID.

Storage Account Details

Choose to configure access to one of the following storage types:

- Azure Data Lake Storage (ADLS). See Azure documentation.
- An Azure Storage Account, known as general or blob storage. See Azure documentation.

The following table describes the information you need to configure Azure Data Lake Storage (ADLS) with the HDInsight cluster:

Property	Description
Azure Data Lake Store Name	Name of the ADLS storage to access. The ADLS storage and the cluster to create must reside in the same region.
Data Lake Service Principal Client ID	A credential that enables programmatic access to ADLS storage. Enables the Informatica domain to communicate with ADLS and run commands and mappings on the HDInsight cluster. The service principal is an Azure user that meets the following requirements: - Permissions to access required directories in ADLS storage. - Certificate-based authentication for ADLS storage. - Key-based authentication for ADLS storage.
Data Lake Service Principal Certificate Contents	The Base64 encoded text of the public certificate used with the service principal. Leave this property blank when you create the cloud provisioning configuration. After you save the cloud provisioning configuration, log in to the VM where the Informatica domain is installed and run infacmd ccps updateADLSCertificate to populate this property.
Data Lake Service Principal Certificate Password	Private key for the service principal. This private key must be associated with the service principal certificate.
Data Lake Service Principal Client Secret	An octet string that provides a key associated with the service principal.
Data Lake Service Principal OAUTH Token Endpoint	Endpoint for OAUTH token based authentication.

The following table describes the information you need to configure Azure General Storage, also known as blob storage, with the HDInsight cluster:

Property	Description
Azure Storage Account Name	Name of the storage account to access. Get the value from the Storage Accounts node in the Azure web console. The storage and the cluster to create must reside in the same region.
Azure Storage Account Key	A key to authenticate access to the storage account. To get the value from the Azure web console, select the storage account, then Access Keys. The console displays the account keys.

Cluster Deployment Details

The following table describes the cluster deployment properties that you configure:

Property	Description
Resource Group	Resource group in which to create the cluster. A resource group is a logical set of Azure resources.
Virtual Network Resource Group	Optional. Resource group to which the virtual network belongs. If you do not specify a resource group, the Data Integration Service assumes that the virtual network is a member of the same resource group as the cluster.
Virtual Network	Name of the virtual network or vnet where you want to create the cluster. Specify a vnet that resides in the resource group that you specified in the Virtual Network Resource Group property. The vnet must be in the same region as the region in which to create the cluster.
Subnet Name	Subnet in which to create the cluster. The subnet must be a part of the vnet that you designated in the previous property. Each vnet can have one or more subnets. The Azure administrator can choose an existing subnet or create one for the cluster.

External Hive Metastore Details

You can specify the properties to enable the cluster to connect to a Hive metastore database that is external to the cluster.

If you do not specify an existing external database in this dialog box, the cluster creates its own database on the cluster. This database is terminated when the cluster is terminated.

You can use an external relational database like MySQL or Amazon RDS as the Hive metastore database. The external database must be on the same cloud platform as the cluster to create.

The following table describes the Hive metastore database properties that you configure:

Property	Description
Database Name	Name of the Hive metastore database.
Database Server Name	Server on which the database resides. **Note:** The database server name on the Azure web console commonly includes the suffix `database.windows.net`. For example: `server123xyz.database.windows.net`. You can specify the database server name without the suffix and Informatica will automatically append the suffix. For example, you can specify `server123xyz`.
Database User Name	User name of the account for the domain to use to access the database.
Database Password	Password for the user account.

Authentication Details

The following table describes authentication properties to configure:

Property	Description
Name	Name of the cloud provisioning configuration.
ID	ID of the cloud provisioning configuration. Default: Same as the cloud provisioning configuration name.
Description	Optional. Description of the cloud provisioning configuration.
Subscription ID	ID of the Azure account to use in the cluster creation process.
Tenant ID	A GUID string associated with the Azure Active Directory.
Client ID	A GUID string that is the same as the Application ID associated with the Service Principal. The Service Principal must be assigned to a role that has permission to create resources in the subscription that you identified in the Subscription ID property.
Client Secret	An octet string that provides a key associated with the client ID.

Storage Account Details

Choose to configure access to one of the following storage types:

- Azure Data Lake Storage (ADLS). See Azure documentation.
- An Azure Storage Account, known as general or blob storage. See Azure documentation.

The following table describes the information you need to configure Azure Data Lake Storage (ADLS) with the HDInsight cluster:

Property	Description
Azure Data Lake Store Name	Name of the ADLS storage to access. The ADLS storage and the cluster to create must reside in the same region.
Data Lake Service Principal Client ID	A credential that enables programmatic access to ADLS storage. Enables the Informatica domain to communicate with ADLS and run commands and mappings on the HDInsight cluster. The service principal is an Azure user that meets the following requirements: - Permissions to access required directories in ADLS storage. - Certificate-based authentication for ADLS storage. - Key-based authentication for ADLS storage.
Data Lake Service Principal Certificate Contents	The Base64 encoded text of the public certificate used with the service principal. Leave this property blank when you create the cloud provisioning configuration. After you save the cloud provisioning configuration, log in to the VM where the Informatica domain is installed and run infacmd ccps updateADLSCertificate to populate this property.
Data Lake Service Principal Certificate Password	Private key for the service principal. This private key must be associated with the service principal certificate.

Property	Description
Data Lake Service Principal Client Secret	An octet string that provides a key associated with the service principal.
Data Lake Service Principal OAUTH Token Endpoint	Endpoint for OAUTH token based authentication.

The following table describes the information you need to configure Azure General Storage, also known as blob storage, with the HDInsight cluster:

Property	Description
Azure Storage Account Name	Name of the storage account to access. Get the value from the Storage Accounts node in the Azure web console. The storage and the cluster to create must reside in the same region.
Azure Storage Account Key	A key to authenticate access to the storage account. To get the value from the Azure web console, select the storage account, then Access Keys. The console displays the account keys.

Cluster Deployment Details

The following table describes the cluster deployment properties that you configure:

Property	Description
Resource Group	Resource group in which to create the cluster. A resource group is a logical set of Azure resources.
Virtual Network Resource Group	Optional. Resource group to which the virtual network belongs. If you do not specify a resource group, the Data Integration Service assumes that the virtual network is a member of the same resource group as the cluster.
Virtual Network	Name of the virtual network or vnet where you want to create the cluster. Specify a vnet that resides in the resource group that you specified in the Virtual Network Resource Group property. The vnet must be in the same region as the region in which to create the cluster.
Subnet Name	Subnet in which to create the cluster. The subnet must be a part of the vnet that you designated in the previous property. Each vnet can have one or more subnets. The Azure administrator can choose an existing subnet or create one for the cluster.

External Hive Metastore Details

You can specify the properties to enable the cluster to connect to a Hive metastore database that is external to the cluster.

You can use an external relational database like MySQL or Amazon RDS as the Hive metastore database. The external database must be on the same cloud platform as the cluster to create.

If you do not specify an existing external database in this dialog box, the cluster creates its own database on the cluster. This database is terminated when the cluster is terminated.

The following table describes the Hive metastore database properties that you configure:

Property	Description
Database Name	Name of the Hive metastore database.
Database Server Name	Server on which the database resides. **Note:** The database server name on the Azure web console commonly includes the suffix `database.windows.net`. For example: `server123xyz.database.windows.net`. You can specify the database server name without the suffix and Informatica will automatically append the suffix. For example, you can specify `server123xyz`.
Database User Name	User name of the account for the domain to use to access the database.
Database Password	Password for the user account.

Databricks Cloud Provisioning Configuration Properties

The properties in the Databricks cloud provisioning configuration enable the Data Integration Service to contact and create resources on the Databricks cloud platform.

The following table describes the Databricks cloud provisioning configuration properties:

Property	Description
Name	Name of the cloud provisioning configuration. **Tip:** Because the Administrator tool lists cloud provisioning configuration objects with other connections, use a naming convention such as "CPC" as part of the name of the object to help identify it.
ID	The cluster ID of the Databricks cluster.
Description	Optional description of the cloud provisioning configuration.
Databricks domain	Domain name of the Databricks deployment.
Databricks token ID	The token ID created within Databricks 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.
Advanced Properties	Advanced properties that are unique to the Databricks cloud provisioning configuration.

Advanced Properties

Configure the following properties in the **Advanced Properties** of the Databricks configuration section:

infaspark.pythontx.exec

Required to run a Python transformation on the Databricks Spark engine. Set to the location of the Python executable binary on the worker nodes in the Databricks cluster.

When you provision the cluster at run time, set this property in the Databricks cloud provisioning configuration. Otherwise, set on the Databricks connection.

For example, set to:

```
infaspark.pythontx.exec=/databricks/python3/bin/python3
```

infaspark.pythontx.executorEnv.PYTHONHOME

Required to run a Python transformation on the Databricks Spark engine. Set to the location of the Python installation directory on the worker nodes in the Databricks cluster.

When you provision the cluster at run time, set this property in the Databricks cloud provisioning configuration. Otherwise, set on the Databricks connection.

For example, set to:

```
infaspark.pythontx.executorEnv.PYTHONHOME=/databricks/python3
```

Amazon Redshift Connection Properties

When you set up an Amazon Redshift connection, you must configure the connection properties.

The following table describes the Amazon Redshift 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 Amazon Redshift in the Database.

The **Details** tab contains the connection attributes of the Amazon Redshift connection. The following table describes the connection attributes:

Property	Description
Username	User name of the Amazon Redshift account.
Password	Password for the Amazon Redshift account.
Access Key ID	Amazon S3 bucket access key ID. **Note:** Required if you do not use AWS Identity and Access Management (IAM) authentication.
Secret Access Key	Amazon S3 bucket secret access key ID. **Note:** Required if you do not use AWS Identity and Access Management (IAM) authentication.

Property	Description
Master Symmetric Key	Optional. Provide a 256-bit AES encryption key in the Base64 format when you enable client-side encryption. You can generate a key using a third-party tool. If you specify a value, ensure that you specify the encryption type as client side encryption in the advanced target properties.
JDBC URL	Amazon Redshift connection URL.
Cluster Region	Optional. The AWS cluster region in which the bucket you want to access resides. Select a cluster region if you choose to provide a custom JDBC URL that does not contain a cluster region name in the **JDBC URL** connection property. If you specify a cluster region in both **Cluster Region** and **JDBC URL** connection properties, the Data Integration Service ignores the cluster region that you specify in the **JDBC URL** connection property. To use the cluster region name that you specify in the **JDBC URL** connection property, select **None** as the cluster region in this property. Select one of the following cluster regions: 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 None. You can only read data from or write data to the cluster regions supported by AWS SDK used by PowerExchange for Amazon Redshift.
Customer Master Key ID	Optional. Specify the customer master key ID generated by AWS Key Management Service (AWS KMS) or the Amazon Resource Name (ARN) of your custom key for cross-account access. You must generate the customer master key corresponding to the region where Amazon S3 bucket resides. You can specify any of the following values: **Customer generated customer master key** Enables client-side or server-side encryption. **Default customer master key** Enables client-side or server-side encryption. Only the administrator user of the account can use the default customer master key ID to enable client-side encryption.

Amazon S3 Connection Properties

When you set up an Amazon S3 connection, you must configure the connection properties.

The following table describes the Amazon S3 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	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 Amazon S3 connection type.
Access Key	Access key to access the Amazon S3 bucket. Provide the access key value based on the following authentication methods: - Basic authentication: provide the actual access key value. - IAM authentication: do not provide the access key value. - Temporary security credentials via assume role: provide access key of an IAM user with no permissions to access Amazon S3 bucket.
Secret Key	Secret access key to access the Amazon S3 bucket. The secret key is associated with the access key and uniquely identifies the account. Provide the access key value based on the following authentication methods: - Basic authentication: provide the actual access secret value. - IAM authentication: do not provide the access secret value. - Temporary security credentials via assume role: provide access secret of an IAM user with no permissions to access Amazon S3 bucket.
IAM Role ARN	The ARN of the IAM role assumed by the user to use the dynamically generated temporary security credentials. Enter the value of this property if you want to use the temporary security credentials to access the AWS resources. If you want to use the temporary security credentials with IAM authentication, do not provide the Access Key and Secret Key connection properties. If you want to use the temporary security credentials without IAM authentication, you must enter the value of the Access Key and Secret Key connection properties. For more information about how to obtain the ARN of the IAM role, see the AWS documentation.
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	Optional. Provide a 256-bit AES encryption key in the Base64 format when you enable client-side encryption. You can generate a master symmetric key using a third-party tool.

Property	Description
S3 Account Type	The type of the Amazon S3 account. Select **Amazon S3 Storage** or **S3 Compatible Storage**. Select the Amazon S3 storage option to use the Amazon S3 services. Select the S3 compatible storage option to specify the endpoint for a third-party storage provider such as Scality RING. By default, Amazon S3 storage is selected.
REST Endpoint	The S3 storage endpoint. Specify the S3 storage endpoint in HTTP/HTTPs format when you select the S3 compatible storage option. For example, `http://s3.isv.scality.com`.
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 (Hong Kong) - 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). Not applicable for S3 compatible storage.
Customer Master Key ID	Optional. Specify the customer master key ID or alias name generated by AWS Key Management Service (AWS KMS) or the Amazon Resource Name (ARN) of your custom key for cross-account access. 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 client-side or server-side encryption. **Default customer master key** Enables client-side or server-side encryption. Only the administrator user of the account can use the default customer master key ID to enable client-side encryption.
Federated SSO IdP	SAML 2.0-enabled identity provider for the federated user single sign-on to use with the AWS account. PowerExchange for Amazon S3 supports only the `ADFS 3.0` identity provider. Select `None` if you do not want to use federated user single sign-on.

Federated user single sign-on connection properties

Configure the following properties when you select `ADFS 3.0` in **Federated SSO IdP**:

Property	Description
Federated User Name	User name of the federated user to access the AWS account through the identity provider.
Federated User Password	Password for the federated user to access the AWS account through the identity provider.
IdP SSO URL	Single sign-on URL of the identity provider for AWS.
SAML Identity Provider ARN	ARN of the SAML identity provider that the AWS administrator created to register the identity provider as a trusted provider.
Role ARN	ARN of the IAM role assumed by the federated user.

Blockchain Connection Properties

When you set up a blockchain connection, you must configure the connection properties.

The following table describes the general connection properties for a blockchain 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.
Swagger File Path	The absolute path of the swagger file path that contains the REST API to communicate with the blockchain. The swagger file must be a JSON file that is stored on the Data Integration Service machine. If the swagger file is in a different file format, such as YAML, convert the file to JSON format.
Base URL	Required. The base URL that is used to access assets on the blockchain.
Auth Type*	Authentication method that the run-time engine uses to connect to the REST server. You can use none, basic, digest, or OAuth.
Auth User ID*	User name to authenticate to the REST server.
Auth Password*	Password for the user name to authenticate to the REST server.

Property	Description
OAuth Consumer Key*	Required for the OAuth authentication type. Client key that is associated with the REST server.
OAuth Consumer Secret*	Required for the OAuth authentication type. Client password to connect to the REST server.
OAuth Token*	Required for the OAuth authentication type. Access token to connect to the REST server.
OAuth Token Secret*	Required for the OAuth authentication type. Password associated with the OAuth token.
Proxy Type*	Type of proxy. You can use no proxy, platform proxy, or custom.
Proxy Details*	Proxy configuration using the format `<host>:<port>`.
TrustStore File Path*	The absolute path of the truststore file that contains the SSL certificate.
TrustStore Password*	Password for the truststore file.
KeyStore File Path*	The absolute path of the keystore file that contains the keys and certificates required to establish a two-way secure connection with the REST server.
KeyStore Password*	Password for the keystore file.
Advanced Properties	List of advanced properties to access an asset on the blockchain. Specify the advanced properties using name-value pairs that are separated by a semicolon. You can use the following advanced properties: - X-API-KEY. Required if you authenticate to the REST server using an API key. The advanced properties that you configure in the connection override the values for the corresponding advanced properties in the blockchain data object. For example, if the connection and the data object both specify a base URL, the value in the connection overrides the value in the data object. **Note:** The advanced properties have the precedence level,**Operation level** > **Object level** > **Connection level**. The properties configured at the operation level will override the properties configured at the object or connection level.
Cookies	Required based on how the REST API is implemented. List of cookie properties to specify the cookie information that is passed to the REST server. Specify the properties using name-value pairs that are separated by a semicolon. The cookie properties that you configure in the connection override the values for the corresponding cookie properties in the blockchain data object.

* The property is ignored. To use the functionality, configure the property as an advanced property and provide a name-value pair based on the property name in the swagger file.

For example, configure the following name-value pair to use basic authorization:

```
Authorization=Basic <credentials>
```

Note: You cannot use **Test Connection** to validate a Blockchain connection.

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**.
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

Property	Description
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.
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*.

Databricks Connection Properties

Use the Databricks connection to run mappings on a Databricks cluster.

A Databricks connection is a cluster type connection. You can create and manage a Databricks connection in the Administrator tool or the Developer tool. You can use infacmd to create a Databricks connection. Configure properties in the Databricks connection to enable communication between the Data Integration Service and the Databricks cluster.

The following table describes the general connection properties for the Databricks 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	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	Optional. The description of the connection. The description cannot exceed 4,000 characters.
Connection Type	Choose Databricks.
Cluster Configuration	Name of the cluster configuration associated with the Databricks environment. Required if you do not configure the cloud provisioning configuration.
Cloud Provisioning Configuration	Name of the cloud provisioning configuration associated with a Databricks cloud platform. Required if you do not configure the cluster configuration.
Staging Directory	The directory where the Databricks Spark engine stages run-time files. If you specify a directory that does not exist, the Data Integration Service creates it at run time. If you do not provide a directory path, the run-time staging files are written to */<cluster staging directory>/DATABRICKS*.
Advanced Properties	List of advanced properties that are unique to the Databricks environment. You can configure run-time properties for the Databricks environment in the Data Integration Service and in the Databricks connection. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Databricks connection. The Data Integration Service processes property overrides based on the following priorities: 1. Databricks connection advanced properties 2. Data Integration Service custom properties **Note:** Informatica does not recommend changing these property values before you consult with third-party documentation, Informatica documentation, or Informatica Global Customer Support. If you change a value without knowledge of the property, you might experience performance degradation or other unexpected results.

Advanced Properties

Configure the following properties in the **Advanced Properties** of the Databricks configuration section:

infaspark.json.parser.mode

Specifies the parser how to handle corrupt JSON records. You can set the value to one of the following modes:

- DROPMALFORMED. The parser ignores all corrupted records. Default mode.
- PERMISSIVE. The parser accepts non-standard fields as nulls in corrupted records.
- FAILFAST. The parser generates an exception when it encounters a corrupted record and the Spark application goes down.

infaspark.json.parser.multiLine

Specifies whether the parser can read a multiline record in a JSON file. You can set the value to true or false. Default is false. Applies only to non-native distributions that use Spark version 2.2.x and above.

infaspark.flatfile.writer.nullValue

When the Databricks Spark engine writes to a target, it converts null values to empty strings (" "). For example, 12, AB,"",23p09udj.

The Databricks Spark engine can write the empty strings to string columns, but when it tries to write an empty string to a non-string column, the mapping fails with a type mismatch.

To allow the Databricks Spark engine to convert the empty strings back to null values and write to the target, configure the property in the Databricks Spark connection.

Set to: TRUE

infaspark.pythontx.exec

Required to run a Python transformation on the Databricks Spark engine. Set to the location of the Python executable binary on the worker nodes in the Databricks cluster.

When you provision the cluster at run time, set this property in the Databricks cloud provisioning configuration. Otherwise, set on the Databricks connection.

For example, set to:

```
infaspark.pythontx.exec=/databricks/python3/bin/python3
```

infaspark.pythontx.executorEnv.PYTHONHOME

Required to run a Python transformation on the Databricks Spark engine. Set to the location of the Python installation directory on the worker nodes in the Databricks cluster.

When you provision the cluster at run time, set this property in the Databricks cloud provisioning configuration. Otherwise, set on the Databricks connection.

For example, set to:

```
infaspark.pythontx.executorEnv.PYTHONHOME=/databricks/python3
```

Google Analytics Connection Properties

When you set up a Google Analytics 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 Analytics 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 Analytics**.
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.
APIVersion	API that PowerExchange for Google Analytics uses to read from Google Analytics reports. Select **Core Reporting API v3**. **Note:** PowerExchange for Google Analytics does not support Analytics Reporting API v4.

Google BigQuery Connection Properties

When you set up a Google BigQuery 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 BigQuery connection properties:

Property	Description
Service Account ID	Specifies the client_email value present in the JSON file that you download after you create a service account in Google BigQuery.
Service Account Key	Specifies the private_key value present in the JSON file that you download after you create a service account in Google BigQuery.

Property	Description
Connection mode	The mode that you want to use to read data from or write data to Google BigQuery. Select one of the following connection modes: - Simple. Flattens each field within the Record data type field as a separate field in the mapping. - Hybrid. Displays all the top-level fields in the Google BigQuery table including Record data type fields. PowerExchange for Google BigQuery displays the top-level Record data type field as a single field of the String data type in the mapping. - Complex. Displays all the columns in the Google BigQuery table as a single field of the String data type in the mapping. Default is Simple.
Schema Definition File Path	Specifies a directory on the client machine where the PowerCenter Integration ServiceData Integration Service must create a JSON file with the sample schema of the Google BigQuery table. The JSON file name is the same as the Google BigQuery table name. Alternatively, you can specify a storage path in Google Cloud Storage where the PowerCenter Integration ServiceData Integration Service must create a JSON file with the sample schema of the Google BigQuery table. You can download the JSON file from the specified storage path in Google Cloud Storage to a local machine.
Project ID	Specifies the project_id value present in the JSON file that you download after you create a service account in Google BigQuery. If you have created multiple projects with the same service account, enter the ID of the project that contains the dataset that you want to connect to.
Storage Path	This property applies when you read or write large volumes of data. Path in Google Cloud Storage where the PowerCenter Integration ServiceData Integration Service creates a local stage file to store the data temporarily. You can either enter the bucket name or the bucket name and folder name. For example, enter `gs://<bucket_name>` or `gs://<bucket_name>/<folder_name>`
Dataset ID	Not applicable for PowerExchange for Google BigQuery.
Use Legacy SQL For Custom Query	Not applicable for PowerExchange for Google BigQuery.
Dataset Name for Custom Query	Not applicable for PowerExchange for Google BigQuery.
Region ID	The region name where the Google BigQuery dataset resides. For example, if you want to connect to a Google BigQuery dataset that resides in Las Vegas region, specify **us-west4** as the **Region ID**. **Note:** In the **Storage Path** connection property, ensure that you specify a bucket name or the bucket name and folder name that resides in the same region as the dataset in Google BigQuery. For more information about the regions supported by Google BigQuery, see the following Google BigQuery documentation:https://cloud.google.com/bigquery/docs/locations

Property	Description
Optional Properties	Specifies whether you can configure certain source and target functionalities through custom properties. You can select one of the following options: - None. Select if you do not want to configure any custom properties. - Required. If you want to specify custom properties to configure the source and target functionalities. Default is None.
Provide Optional Properties	Comma-separated key-value pairs of custom properties to enable certain source and target functionalities. Appears only when you select **Required** in the Optional Properties. For more information about the list of custom properties that you can specify, see the Informatica Knowledge Base article: https://kb.informatica.com/faq/7/Pages/26/632722.aspx

Google Cloud Spanner Connection Properties

When you set up a Google Cloud Spanner 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 Spanner 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 Spanner.
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.

Property	Description
Service Account Key	Specifies the private_key value present in the JSON file that you download after you create a service account.
Instance ID	Name of the instance that you created in Google Cloud Spanner.

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.

Hadoop Connection Properties

Use the Hadoop connection to configure mappings to run on a Hadoop cluster. A Hadoop connection is a cluster type connection. You can create and manage a Hadoop connection in the Administrator tool or the Developer tool. You can use infacmd to create a Hadoop connection. Hadoop connection properties are case sensitive unless otherwise noted.

You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities:

1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option
2. Mapping run-time properties for the Hadoop environment

3. Hadoop connection advanced properties for run-time engines
4. Hadoop connection advanced general properties, environment variables, and classpaths
5. Data Integration Service custom properties

Note: When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez`

Hadoop Cluster Properties

Configure properties in the Hadoop connection to enable communication between the Data Integration Service and the Hadoop cluster.

The following table describes the general connection properties for the Hadoop 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	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.
Cluster Configuration	The name of the cluster configuration associated with the Hadoop environment. Required if you do not configure the Cloud Provisioning Configuration.
Cloud Provisioning Configuration	Name of the cloud provisioning configuration associated with a cloud platform such as Amazon AWS or Microsoft Azure. Required if you do not configure the Cluster Configuration.

Property	Description
Cluster Environment Variables*	Environment variables that the Hadoop cluster uses. If you use a Cloudera CDH 6.x cluster or a Cloudera CDP cluster, configure the locale setting as cluster environment variables. In Cloudera Manager, you must also add the environment variables to the following YARN property: `yarn.nodemanager.env-whitelist` For example, the variable ORACLE_HOME represents the directory where the Oracle database client software is installed. You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities: 1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option 2. Mapping run-time properties for the Hadoop environment 3. Hadoop connection advanced properties for run-time engines 4. Hadoop connection advanced general properties, environment variables, and classpaths 5. Data Integration Service custom properties **Note:** When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez`
Cluster Library Path*	The path for shared libraries on the cluster. The $DEFAULT_CLUSTER_LIBRARY_PATH variable contains a list of default directories.
Cluster Classpath*	The classpath to access the Hadoop jar files and the required libraries. The $DEFAULT_CLUSTER_CLASSPATH variable contains a list of paths to the default jar files and libraries. You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities: 1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option 2. Mapping run-time properties for the Hadoop environment 3. Hadoop connection advanced properties for run-time engines 4. Hadoop connection advanced general properties, environment variables, and classpaths 5. Data Integration Service custom properties **Note:** When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez`

Property	Description
Cluster Executable Path*	The path for executable files on the cluster. The $DEFAULT_CLUSTER_EXEC_PATH variable contains a list of paths to the default executable files.
* Informatica does not recommend changing these property values before you consult with third-party documentation, Informatica documentation, or Informatica Global Customer Support. If you change a value without knowledge of the property, you might experience performance degradation or other unexpected results.	

Common Properties

The following table describes the common connection properties that you configure for the Hadoop connection:

Property	Description
Impersonation User Name	Required if the Hadoop cluster uses Kerberos authentication. Hadoop impersonation user. The user name that the Data Integration Service impersonates to run mappings in the Hadoop environment. Data Engineering Integration supports operating system profiles on all Hadoop distributions. In the Hadoop run-time environment, the Data Integration Service pushes the processing to the Hadoop cluster and the run-time engines run mappings with the operating system profile specified Hadoop impersonation properties.
Temporary Table Compression Codec	Hadoop compression library for a compression codec class name. **Note:** The Spark engine does not support compression settings for temporary tables. When you run mappings on the Spark engine, the Spark engine stores temporary tables in an uncompressed file format.
Codec Class Name	Codec class name that enables data compression and improves performance on temporary staging tables.
Hive Staging Database Name	Namespace for Hive staging tables. Use the name `default` for tables that do not have a specified database name. If you do not configure a namespace, the Data Integration Service uses the Hive database name in the Hive target connection to create staging tables. When you run a mapping in the native environment to write data to Hive, you must configure the Hive staging database name in the Hive connection. The Data Integration Service ignores the value you configure in the Hadoop connection.
Environment SQL	SQL commands to set the Hadoop environment. The Data Integration Service executes the environment SQL at the beginning of each Hive script generated by a HiveServer2 job. The following rules and guidelines apply to the usage of environment SQL: - You can use environment SQL to define Hadoop or Hive parameters that you want to use in the PreSQL commands or in custom queries. - If you use multiple values for the Environment SQL property, ensure that there is no space between the values.

Property	Description
Engine Type	The Data Integration Service uses HiveServer2 to process portions of some jobs by running HiveServer2 tasks on the Spark engine. When you import the cluster configuration through the Administrator tool, you can choose to create connections. The engine type property is populated by default based on the distribution. When you manually create a connection, you must configure the engine type. You can specify the engine type based on the following Hadoop distributions: - Amazon EMR. Tez - Azure HDI. Tez - Cloudera CDH. MRv2 - Cloudera CDP. Tez - Dataproc. MRv2 - Hortonworks HDP. Tez - MapR. MRv2
Advanced Properties	List of advanced properties that are unique to the Hadoop environment. The properties are common to the Blaze and Spark engines. The advanced properties include a list of default properties. You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities: 1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option 2. Mapping run-time properties for the Hadoop environment 3. Hadoop connection advanced properties for run-time engines 4. Hadoop connection advanced general properties, environment variables, and classpaths 5. Data Integration Service custom properties **Note:** When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez` **Note:** Informatica does not recommend changing these property values before you consult with third-party documentation, Informatica documentation, or Informatica Global Customer Support. If you change a value without knowledge of the property, you might experience performance degradation or other unexpected results.

Reject Directory Properties

The following table describes the connection properties that you configure to the Hadoop Reject Directory.

Property	Description
Write Reject Files to Hadoop	If you use the Blaze engine to run mappings, select the check box to specify a location to move reject files. If checked, the Data Integration Service moves the reject files to the HDFS location listed in the property, Reject File Directory. By default, the Data Integration Service stores the reject files based on the RejectDir system parameter.
Reject File Directory	The directory for Hadoop mapping files on HDFS when you run mappings.

Blaze Configuration

The following table describes the connection properties that you configure for the Blaze engine:

Property	Description
Blaze Staging Directory	The HDFS file path of the directory that the Blaze engine uses to store temporary files. Verify that the directory exists. The YARN user, Blaze engine user, and mapping impersonation user must have write permission on this directory. Default is `/blaze/workdir`. If you clear this property, the staging files are written to the Hadoop staging directory `/tmp/blaze_<user name>`.
Blaze User Name	The owner of the Blaze service and Blaze service logs. When the Hadoop cluster uses Kerberos authentication, the default user is the Data Integration Service SPN user. When the Hadoop cluster does not use Kerberos authentication and the Blaze user is not configured, the default user is the Data Integration Service user.
Minimum Port	The minimum value for the port number range for the Blaze engine. Default is 12300.
Maximum Port	The maximum value for the port number range for the Blaze engine. Default is 12600.
YARN Queue Name	The YARN scheduler queue name used by the Blaze engine that specifies available resources on a cluster. **Note:** If YARN preemption is enabled on the cluster, verify with the Hadoop administrator that preemption is disabled on the queue associated with the Blaze engine.
Blaze Job Monitor Address	The host name and port number for the Blaze Job Monitor. Use the following format: `<hostname>:<port>` Where - <hostname> is the host name or IP address of the Blaze Job Monitor server. - <port> is the port on which the Blaze Job Monitor listens for remote procedure calls (RPC). For example, enter: `myhostname:9080`

Property	Description
Blaze YARN Node Label	Node label that determines the node on the Hadoop cluster where the Blaze engine runs. If you do not specify a node label, the Blaze engine runs on the nodes in the default partition. If the Hadoop cluster supports logical operators for node labels, you can specify a list of node labels. To list the node labels, use the operators `&&` (AND), `||` (OR), and `!` (NOT). **Note:** You cannot use node labels on a Cloudera CDH cluster.
Advanced Properties	List of advanced properties that are unique to the Blaze engine. The advanced properties include a list of default properties. You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities: 1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option 2. Mapping run-time properties for the Hadoop environment 3. Hadoop connection advanced properties for run-time engines 4. Hadoop connection advanced general properties, environment variables, and classpaths 5. Data Integration Service custom properties **Note:** When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez` **Note:** Informatica does not recommend changing these property values before you consult with third-party documentation, Informatica documentation, or Informatica Global Customer Support. If you change a value without knowledge of the property, you might experience performance degradation or other unexpected results.

Spark Configuration

The following table describes the connection properties that you configure for the Spark engine:

Property	Description
Spark Staging Directory	The HDFS file path of the directory that the Spark engine uses to store temporary files for running jobs. The YARN user, Data Integration Service user, and mapping impersonation user must have write permission on this directory. If you do not specify a file path, by default, the temporary files are written to the Hadoop staging directory `/tmp/SPARK_<user name>`. When you run Sqoop jobs on the Spark engine, the Data Integration Service creates a Sqoop staging directory within the Spark staging directory to store temporary files: `<Spark staging directory>/sqoop_staging`
Spark Event Log Directory	Optional. The HDFS file path of the directory that the Spark engine uses to log events.

Property	Description
YARN Queue Name	The YARN scheduler queue name used by the Spark engine that specifies available resources on a cluster. The name is case sensitive.
Advanced Properties	List of advanced properties that are unique to the Spark engine. The advanced properties include a list of default properties. You can configure run-time properties for the Hadoop environment in the Data Integration Service, the Hadoop connection, and in the mapping. You can override a property configured at a high level by setting the value at a lower level. For example, if you configure a property in the Data Integration Service custom properties, you can override it in the Hadoop connection or in the mapping. The Data Integration Service processes property overrides based on the following priorities: 1. Mapping custom properties set using infacmd ms runMapping with the `-cp` option 2. Mapping run-time properties for the Hadoop environment 3. Hadoop connection advanced properties for run-time engines 4. Hadoop connection advanced general properties, environment variables, and classpaths 5. Data Integration Service custom properties **Note:** When a mapping uses Hive Server 2 to run a job or parts of a job, you cannot override properties that are configured on the cluster level in preSQL or post-SQL queries or SQL override statements. Workaround: Instead of attempting to use the cluster configuration on the domain to override cluster properties, pass the override settings to the JDBC URL. For example: `beeline -u "jdbc:hive2://<domain host>:<port_number>/tpch_text_100" --hiveconf hive.execution.engine=tez` **Note:** Informatica does not recommend changing these property values before you consult with third-party documentation, Informatica documentation, or Informatica Global Customer Support. If you change a value without knowledge of the property, you might experience performance degradation or other unexpected results.

HDFS Connection Properties

Use a Hadoop File System (HDFS) connection to access data in the Hadoop cluster. The HDFS connection is a file system type connection. You can create and manage an HDFS connection in the Administrator tool, Analyst tool, or the Developer tool. HDFS connection properties are case sensitive unless otherwise noted.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes HDFS connection properties:

Property	Description
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.
Location	The domain where you want to create the connection. Not valid for the Analyst tool.

Property	Description
Type	The connection type. Default is Hadoop File System.
User Name	User name to access HDFS.
NameNode URI	The URI to access the storage system. You can find the value for `fs.defaultFS` in the `core-site.xml` configuration set of the cluster configuration. If you create connections when you import the cluster configuration, the NameNode URI property is populated by default, and it is updated each time you refresh the cluster configuration. If you use a Cloudera CDP Public Cloud compute cluster and the HDFS is on a Cloudera Data Lake cluster, set the property *spark.yarn.access.hadoopFileSystems* in the Spark properties of the Hadoop Connection to the same value as set here.

Accessing Multiple Storage Types

Use the NameNode URI property in the connection parameters to connect to various storage types. The following table lists the storage type and the NameNode URI format for the storage type:

Storage	NameNode URI Format
HDFS	`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). `hdfs://<nameservice>` in case of NameNode high availability.
MapR-FS	`maprfs:///`

WASB in HDInsight	`wasb://<container_name>@<account_name>.blob.core.windows.net/<path>`
	where:
	- `<container_name>` identifies a specific Azure Storage Blob container.
	Note: `<container_name>` is optional.
	- `<account_name>` identifies the Azure Storage Blob object.
	Example:
	`wasb://infabdmoffering1storage.blob.core.windows.net/infabdmoffering1cluster/mr-history`

ADLS in HDInsight	`adl://home`

When you create a cluster configuration from an Azure HDInsight cluster, the cluster configuration uses either ADLS or WASB as the primary storage. You cannot create a cluster configuration with ADLS or WASB as the secondary storage. You can edit the NameNode URI property in the HDFS connection to connect to a local HDFS location.

HBase Connection Properties

Use an HBase connection to access HBase. The HBase connection is a NoSQL connection. You can create and manage an HBase connection in the Administrator tool or the Developer tool. HBase connection properties are case sensitive unless otherwise noted.

The following table describes HBase 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 HBase.
Database Type	Type of database that you want to connect to. Select **HBase** to create a connection for an HBase table.

HBase Connection Properties for MapR-DB

Use an HBase connection to connect to a MapR-DB table. The HBase connection is a NoSQL connection. You can create and manage an HBase connection in the Administrator tool or the Developer tool. HBase connection properties are case sensitive unless otherwise noted.

The following table describes the HBase connection properties for MapR-DB:

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	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	Description of the connection. The description cannot exceed 4,000 characters.
Location	Domain where you want to create the connection.
Type	Connection type. Select **HBase**.
Database Type	Type of database that you want to connect to. Select **MapR-DB** to create a connection for a MapR-DB table.
Cluster Configuration	The name of the cluster configuration associated with the Hadoop environment.
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`

Hive Connection Properties

Use the Hive connection to access Hive data. A Hive connection is a database type connection. You can create and manage a Hive connection in the Administrator tool, Analyst tool, or the Developer tool. Hive connection properties are case sensitive unless otherwise noted.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes Hive 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 4000 characters.

Property	Description
Location	The domain where you want to create the connection. Not valid for the Analyst tool.
Type	The connection type. Select Hive.
LDAP username	LDAP user name of the user that the Data Integration Service impersonates to run mappings on a Hadoop cluster. The user name depends on the JDBC connection string that you specify in the Metadata Connection String or Data Access Connection String for the native environment. If the Hadoop cluster uses Kerberos authentication, the principal name for the JDBC connection string and the user name must be the same. Otherwise, the user name depends on the behavior of the JDBC driver. With Hive JDBC driver, you can specify a user name in many ways and the user name can become a part of the JDBC URL. If the Hadoop cluster does not use Kerberos authentication, the user name depends on the behavior of the JDBC driver. If you do not specify a user name, the Hadoop cluster authenticates jobs based on the following criteria: - The Hadoop cluster does not use Kerberos authentication. It authenticates jobs based on the operating system profile user name of the machine that runs the Data Integration Service. - The Hadoop cluster uses Kerberos authentication. It authenticates jobs based on the SPN of the Data Integration Service. LDAP username will be ignored.
Password	Password for the LDAP username.
Environment SQL	SQL commands to set the Hadoop environment. In native environment type, the Data Integration Service executes the environment SQL each time it creates a connection to a Hive metastore. If you use the Hive connection to run profiles on a Hadoop cluster, the Data Integration Service executes the environment SQL at the beginning of each Hive session. The following rules and guidelines apply to the usage of environment SQL in both connection modes: - Use the environment SQL to specify Hive queries. - Use the environment SQL to set the classpath for Hive user-defined functions and then use environment SQL or PreSQL to specify the Hive user-defined functions. You cannot use PreSQL in the data object properties to specify the classpath. If you use Hive user-defined functions, you must copy the .jar files to the following directory: `<Informatica installation directory>/services/shared/hadoop/<Hadoop distribution name>/extras/hive-auxjars` - You can use environment SQL to define Hadoop or Hive parameters that you want to use in the PreSQL commands or in custom queries. - If you use multiple values for the Environment SQL property, ensure that there is no space between the values.
SQL Identifier Character	The type of character used to identify special characters and reserved SQL keywords, such as WHERE. The Data Integration Service places the selected character around special characters and reserved SQL keywords. The Data Integration Service also uses this character for the **Support mixed-case identifiers** property.

Properties to Access Hive as Source or Target

The following table describes the connection properties that you configure to access Hive as a source or target:

Property	Description
JDBC Driver Class Name	Name of the Hive JDBC driver class. If you leave this option blank, the Developer tool uses the default Apache Hive JDBC driver shipped with the distribution. If the default Apache Hive JDBC driver does not fit your requirements, you can override the Apache Hive JDBC driver with a third-party Hive JDBC driver by specifying the driver class name.
Metadata Connection String	The JDBC connection URI used to access the metadata from the Hadoop server. You can use PowerExchange for Hive to communicate with a HiveServer service or HiveServer2 service. To connect to HiveServer, specify the connection string in the following format: `jdbc:hive2://<hostname>:<port>/<db>` Where - <hostname> is name or IP address of the machine on which HiveServer2 runs. - <port> is the port number on which HiveServer2 listens. - <db> is the database name to which you want to connect. If you do not provide the database name, the Data Integration Service uses the default database details. To connect to HiveServer2, use the connection string format that Apache Hive implements for that specific Hadoop Distribution. For more information about Apache Hive connection string formats, see the Apache Hive documentation. For user impersonation, you must add `hive.server2.proxy.user=<xyz>` to the JDBC connection URI. If you do not configure user impersonation, the current user's credentials are used connect to the HiveServer2. If the Hadoop cluster uses SSL or TLS authentication, you must add `ssl=true` to the JDBC connection URI. For example: `jdbc:hive2://<hostname>:<port>/<db>;ssl=true` If you use self-signed certificate for SSL or TLS authentication, ensure that the certificate file is available on the client machine and the Data Integration Service machine. For more information, see the*Data Engineering Integration Guide*.
Bypass Hive JDBC Server	JDBC driver mode. Select the check box to use the embedded JDBC driver mode. To use the JDBC embedded mode, perform the following tasks: - Verify that Hive client and Informatica services are installed on the same machine. - Configure the Hive connection properties to run mappings on a Hadoop cluster. If you choose the non-embedded mode, you must configure the Data Access Connection String. Informatica recommends that you use the JDBC embedded mode.
Fine Grained Authorization	When you select the option to observe fine grained authorization in a Hive source, the mapping observes the following: - Row and column level restrictions. Applies to Hadoop clusters where Sentry or Ranger security modes are enabled. - Data masking rules. Applies to masking rules set on columns containing sensitive data by Dynamic Data Masking. If you do not select the option, the Blaze and Spark engines ignore the restrictions and masking rules, and results include restricted or sensitive data.

Property	Description
Data Access Connection String	The connection string to access data from the Hadoop data store. To connect to HiveServer, specify the non-embedded JDBC mode connection string in the following format: `jdbc:hive2://<hostname>:<port>/<db>` Where - <hostname> is name or IP address of the machine on which HiveServer2 runs. - <port> is the port number on which HiveServer2 listens. - <db> is the database to which you want to connect. If you do not provide the database name, the Data Integration Service uses the default database details. To connect to HiveServer2, use the connection string format that Apache Hive implements for the specific Hadoop Distribution. For more information about Apache Hive connection string formats, see the Apache Hive documentation. For user impersonation, you must add `hive.server2.proxy.user=<xyz>` to the JDBC connection URI. If you do not configure user impersonation, the current user's credentials are used connect to the HiveServer2. If the Hadoop cluster uses SSL or TLS authentication, you must add `ssl=true` to the JDBC connection URI. For example: `jdbc:hive2://<hostname>:<port>/<db>;ssl=true` If you use self-signed certificate for SSL or TLS authentication, ensure that the certificate file is available on the client machine and the Data Integration Service machine. For more information, see the *Data Engineering Integration Guide*.
Hive Staging Directory on HDFS	HDFS directory for Hive staging tables. You must grant execute permission to the Hadoop impersonation user and the mapping impersonation users. This option is applicable and required when you write data to a Hive target in the native environment.
Hive Staging Database Name	Namespace for Hive staging tables. The Hive Staging Database Name is automatically updated from the Data Access Connection String. If you want to override the default name, you need to configure the Hive Staging Database Name in the Hive connection. This option is applicable when you run a mapping in the native environment to write data to a Hive target. If you run the mapping on the Blaze or Spark engine, you do not need to configure the Hive staging database name in the Hive connection. The Data Integration Service uses the value that you configure in the Hadoop connection.

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 Connection String

Specify the connection string in the JDBC connection to connect to the database.

Specify the connection string in the following format:

```
jdbc:<subprotocol>:<subname>
```

Use the following sample connection strings that you can enter for the applicable database type:

- DataDirect Oracle JDBC driver:
  ```
  jdbc:informatica:oracle://<host>:<port>;SID=<value>
  ```
- Oracle JDBC driver:
  ```
  jdbc:oracle:thin:@//<host>:<port>:<SID>
  ```
- DataDirect IBM DB2 JDBC driver:
  ```
  jdbc:informatica:db2://<host>:<port>;DatabaseName=<value>
  ```
- IBM DB2 JDBC driver:
  ```
  jdbc:db2://<host>:<port>/<database_name>
  ```
- DataDirect Microsoft SQL Server JDBC driver:
  ```
  jdbc:informatica:sqlserver://<host>;DatabaseName=<value>
  ```
- Microsoft SQL Server JDBC driver:
  ```
  jdbc:sqlserver://<host>;DatabaseName=<value>
  ```
- Netezza JDBC driver:
  ```
  jdbc:netezza://<host>:<port>/<database_name>
  ```
- Pivotal Greenplum driver:
  ```
  jdbc:pivotal:greenplum://<host>:<port>;/database_name=<value>
  ```
- Postgres Greenplum driver:
  ```
  jdbc:postgressql://<host>:<port>/<database_name>
  ```
- Teradata JDBC driver:
  ```
  jdbc:teradata://<host>/database_name=<value>,tmode=<value>,charset=<value>
  ```
- JDBC driver for Delta Lake:
  ```
  jdbc:spark://<host name>:443/default;transportMode=http;ssl=1;httpPath=sql/
  protocolv1/o/<cluster ID>;AuthMech=3;UID=token;PWD=<token string>
  ```

Use the following sample connection strings that you can enter for an SSL-enabled applicable database type:

- DataDirect Oracle JDBC driver:
  ```
  jdbc:informatica:oracle://
  <host_name>:<port>;CatalogOptions=6;ServiceName=<service_name>;trustStorePassword=<tru
  ststore_password>;
  keyStorePassword=<keystore_password>;CryptoProtocolVersion=TLSv1.2;keyStore=<keystore_
  location_of_ewallet.p12_file>;
  trustStore=<truststore_location_of_truststore.p12_file>;HostNameInCertificate=<databas
  e_host_name>;encryptionMethod=SSL;
  ValidateServerCertificate=True;
  ```
- Oracle JDBC driver:
  ```
  jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=<host>)
  (PORT=<port_number>))
  (CONNECT_DATA=(SERVICE_NAME=<service_name>)))"
  ```
- DataDirect Microsoft SQL Server JDBC driver:
  ```
  jdbc:informatica:sqlserver://
  <host_name>:<port>;databaseName=<database_name>;EncryptionMethod=SSL;
  CryptoProtocolVersion=<TLSv1.2_or_TLSv1.1_or_TLSv1>;ValidateServerCertificate=false;Tr
  ustStore=<truststore_location>;
  TrustStorePassword=<truststore_password>
  ```
- Microsoft SQL Server JDBC driver:
  ```
  jdbc:sqlserver://
  <host_name>:<port>;databaseName=<database_name>;integratedSecurity=false;encrypt=true;
  trustServerCertificate=true;
  TrustStore=/
  <truststore_location>;TrustStorePassword=<truststore_password>;user=<user_name>;passwo
  rd=<password>
  ```

For more information about the connection string to use with specific drivers, see the vendor documentation.

Sqoop Connection-Level Arguments

In the JDBC connection, you can define the arguments that Sqoop must use to connect to the database. The Data Integration Service merges the arguments that you specify with the default command that it constructs based on the JDBC connection properties. The arguments that you specify take precedence over the JDBC connection properties.

If you want to use the same driver to import metadata and run the mapping, and do not want to specify any additional Sqoop arguments, select **Sqoop v1.x** from the **Use Sqoop Version** list and leave the **Sqoop Arguments** field empty in the JDBC connection. The Data Integration Service constructs the Sqoop command based on the JDBC connection properties that you specify.

However, if you want to use a different driver for run-time tasks or specify additional run-time Sqoop arguments, select **Sqoop v1.x** from the **Use Sqoop Version** list and specify the arguments in the **Sqoop Arguments** field.

A mapping that contains an Update Strategy transformation cannot use a Sqoop-enabled JDBC connection to write to a target. To run the mapping, disable the Sqoop connector in the Write transformation.

You can configure the following Sqoop arguments in the JDBC connection:

driver

Defines the JDBC driver class that Sqoop must use to connect to the database.

Use the following syntax:

```
--driver <JDBC driver class>
```

For example, use the following syntax depending on the database type that you want to connect to:

- Aurora: `--driver com.mysql.jdbc.Driver`
- Greenplum: `--driver org.postgresql.Driver`
- IBM DB2: `--driver com.ibm.db2.jcc.DB2Driver`
- IBM DB2 z/OS: `--driver com.ibm.db2.jcc.DB2Driver`
- Microsoft SQL Server: `--driver com.microsoft.sqlserver.jdbc.SQLServerDriver`
- Netezza: `--driver org.netezza.Driver`
- Oracle: `--driver oracle.jdbc.driver.OracleDriver`
- Teradata: `--driver com.teradata.jdbc.TeraDriver`

connect

Defines the JDBC connection string that Sqoop must use to connect to the database. The JDBC connection string must be based on the driver that you define in the driver argument.

Use the following syntax:

```
--connect <JDBC connection string>
```

For example, use the following syntax depending on the database type that you want to connect to:

- Aurora: `--connect "jdbc:mysql://<host_name>:<port>/<schema_name>"`
- Greenplum: `--connect jdbc:postgresql://<host_name>:<port>/<database_name>`
- IBM DB2: `--connect jdbc:db2://<host_name>:<port>/<database_name>`
- IBM DB2 z/OS: `--connect jdbc:db2://<host_name>:<port>/<database_name>`

- Microsoft SQL Server: `--connect jdbc:sqlserver://<host_name>:<port or named_instance>;databaseName=<database_name>`
- Netezza: `--connect "jdbc:netezza://<database_server_name>:<port>/<database_name>;schema=<schema_name>"`
- Oracle: `--connect jdbc:oracle:thin:@<database_host_name>:<database_port>:<database_SID>`
- Teradata: `--connect jdbc:teradata://<host_name>/database=<database_name>`

Use the following syntax to connect to an SSL-enabled database:

```
--connect <JDBC connection string>
```

For example, use the following syntax depending on the database type that you want to connect to:

- Microsoft SQL Server: `--connect jdbc:sqlserver://<host_name>:<port>;databaseName=<database_name>;integratedSecurity=false;encrypt=true;trustServerCertificate=true;TrustStore=/<truststore_location>;TrustStorePassword=<truststore_password>;user=<user_name>;password=<password>`
- Oracle: `--connect jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=<host>)(PORT=<port_number>))(CONNECT_DATA=(SERVICE_NAME=<service_name>)))"`

connection-param-file

Defines the extra JDBC parameters through a property file that Sqoop must use to connect to the database. The contents of this file are parsed as standard Java properties and passed into the driver when you create a connection.

Use the following syntax:

```
--connection-param-file <parameter file name>
```

For example, use the following syntax to use the parameter file when you connect to the Oracle database.

```
--connection-param-file param_file
```

connection-manager

Defines the connection manager class name that Sqoop must use to connect to the database.

Use the following syntax:

```
--connection-manager <connection manager class name>
```

For example, use the following syntax to use the generic JDBC manager class name:

```
--connection-manager org.apache.sqoop.manager.GenericJdbcManager
```

direct

When you read data from or write data to Oracle, you can configure the direct argument to enable Sqoop to use OraOop. OraOop is a specialized Sqoop plug-in for Oracle that uses native protocols to connect to the Oracle database. When you configure OraOop, the performance improves.

You can configure OraOop when you run Sqoop mappings on the Spark engine.

Use the following syntax:

```
--direct
```

When you use OraOop, you must use the following syntax to specify multiple arguments:

```
-D<argument=value> -D<argument=value>
```

Note: If you specify multiple arguments and include a space character between -D and the argument name-value pair, Sqoop considers only the first argument and ignores the remaining arguments.

If you do not direct the job to a specific queue, the Spark engine uses the default queue.

-Dsqoop.connection.factories

To run the mapping on the Blaze engine with the Teradata Connector for Hadoop (TDCH) specialized connectors for Sqoop, you must configure the -Dsqoop.connection.factories argument. Use the argument to define the TDCH connection factory class that Sqoop must use. 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 -Dsqoop.connection.factories argument as follows:
  ```
  -Dsqoop.connection.factories=com.cloudera.connector.teradata.TeradataManagerFactory
  ```
- To use Hortonworks Connector for Teradata (powered by the Teradata Connector for Hadoop), configure the -Dsqoop.connection.factories argument as follows:
  ```
  -Dsqoop.connection.factories=org.apache.sqoop.teradata.TeradataManagerFactory
  ```

Note: To run the mapping on the Spark engine, you do not need to configure the -Dsqoop.connection.factories argument. The Data Integration Service invokes Cloudera Connector Powered by Teradata and Hortonworks Connector for Teradata (powered by the Teradata Connector for Hadoop) by default.

--infaoptimize

Use this argument to disable the performance optimization of Sqoop pass-through mappings on the Spark engine.

When you run a Sqoop pass-through mapping on the Spark engine, the Data Integration Service optimizes mapping performance in the following scenarios:

- You read data from a Sqoop source and write data to a Hive target that uses the Text format.
- You read data from a Sqoop source and write data to an HDFS target that uses the Flat, Avro, or Parquet format.

If you want to disable the performance optimization, set the --infaoptimize argument to false. For example, if you see data type issues after you run an optimized Sqoop mapping, you can disable the performance optimization.

Use the following syntax:

```
--infaoptimize false
```

For a complete list of the Sqoop arguments that you can configure, see the Sqoop documentation.

Delta Lake JDBC Connection Properties

To enable the domain to access Delta Lake resources in the cloud platform environment, you must manually create and configure a JDBC connection.

Configure the following connection properties:

Property	Description
Name	Type a name for the connection. Example: DatabricksDeltaCxn
ID	Default: Automatically populated with the connection name. Changing this default value is optional.
Username	Type the following value to use the Databricks user token: `token`
Password	Value of the token that you configured for the Databricks user.
JDBC Driver Class Name	Type the following value: `com.simba.spark.jdbc4.Driver`
Connection String	Connection to the Delta Lake resource. This connection string contains all the information that the domain needs to connect to the resource. The connection string contains the following elements: - jdbc:spark://<server host name>. - Port number. - Transport mode. - ssl. Use `1` to enable SSL. - httpPath. - UID. User ID that will be used to run jobs on the cluster. Use `token`. - PWD. Value of the token that you configured for the Databricks user. Example: `jdbc:spark://westus.azuredatabricks.net:443/default;transportMode=http;ssl=1;httpPath=sql/protocolv1/o/1654523072521724/0123-456789-films259;AuthMech=3;UID=token;PWD=<token string>` To get the value of these parameters from the Advanced Options area of the cluster configuration settings: 1. In the Databricks environment, select Clusters. 2. Select the cluster to connect to. 3. Expand the Advanced Options and click the **JDBC/ODBC** tab. For more information about the JDBC connection string for Databricks, see the Databricks documentation.

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.
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.

Property	Description
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 Kafka as a message target. You can create and manage a Kafka connection in the Developer tool or through infacmd.

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.
Type	The connection type.

The following table describes the Kafka broker properties for the Kafka connection:

Property	Description
Kafka Broker List	Comma-separated list of Kafka brokers which maintains the configuration of the Kafka messaging broker. To specify a Kafka broker, use the following format: <IP Address>:<port>
ZooKeeper Host Port List	Optional. Comma-separated list of Apache ZooKeeper which maintains the configuration of the Kafka messaging broker. To specify the ZooKeeper, use the following format: <IP Address>:<port>
Retry Timeout	Number of seconds the Integration Service attempts to reconnect to the Kafka broker to 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	Configure the Kafka messaging broker version to 0.10.1.x-2.0.0.

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.
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.
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.

Microsoft Azure Blob Storage Connection Properties

Use a Microsoft Azure SQL Blob Storage connection to access a Microsoft Azure Blob Storage.

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 Blob Storage connection in the Administrator tool or the Developer tool. The following table describes the Microsoft Azure Blob Storage connection properties:

Property	Description
Name	Name of the Microsoft Azure Blob Storage connection.
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	Description of the connection.

Property	Description
Location	The domain where you want to create the connection.
Type	Type of connection. Select Azure Blob Storage.

The **Connection Details** tab contains the connection attributes of the Microsoft Azure Blob Storage connection. The following table describes the connection attributes:

Property	Description
Account Name	Name of the Microsoft Azure Storage account.
Authorization Type	Authorization type. You can select any of the following authorization mechanisms: - Shared Key Authorization - Shared Access Signatures
Account Key	Microsoft Azure Storage access key. Applies when you select shared key authorization.
SAS Token	SAS URI with SAS token that you generate on Microsoft Azure portal for your account. Applies when you select shared access signature authorization type. **Note:** You must provide a valid SAS URI with a valid SAS token.
Container Name	The root container or sub-folders with the absolute path. **Note:** To import complex files, specify only the root container.
Endpoint Suffix	Type of Microsoft Azure end-points. You can select any of the following end-points: - `core.windows.net`: Default - `core.usgovcloudapi.net`: To select the US government Microsoft Azure end-points - `core.chinacloudapi.cn`: Not applicable

Microsoft Azure Cosmos DB SQL API Connection Properties

Use a Microsoft Azure Cosmos DB connection to connect to the Cosmos DB database. When you create a Microsoft Azure Cosmos DB connection, you enter information for metadata and data access.

The following table describes the Microsoft Azure Cosmos DB connection properties:

Property	Description
Name	Name of the Cosmos DB connection.
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.

Property	Description
Description	Description of the connection. The description cannot exceed 765 characters.
Location	The project or folder in the Model repository where you want to store the Cosmos DB connection.
Type	Select Microsoft Azure Cosmos DB SQL API.
Cosmos DB URI	The URI of Microsoft Azure Cosmos DB account.
Key	The primary and secondary key to which provides you complete administrative access to the resources within Microsoft Azure Cosmos DB account.
Database	Name of the database that contains the collections from which you want to read or write JSON documents.

Note: You can find the Cosmos DB URI and Key values in the **Keys** settings on Azure portal. Contact your Azure administrator for more details.

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 Properties

Use a Microsoft Azure Data Lake Storage Gen2 connection to access a Microsoft Azure Data Lake Storage Gen2.

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 Gen2 connection in the Administrator tool or the Developer tool. The following table describes the Microsoft Azure Data Lake Storage Gen2 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 Gen2.

The following table describes the properties for metadata access:

Property	Description
Account Name	The Microsoft Azure Data Lake Storage Gen2 account name or the service name.
Authentication Type	Authentication type to access the Microsoft Azure Data Lake Storage Gen2 account. Select one of the following options: - **Service Principal Authentication**. Select to use the client ID, client secret, and tenant ID to connect to Microsoft Azure Data Lake Storage Gen2. - **Shared Key Authentication**. Select to use the account key to connect to Microsoft Azure Data Lake Storage Gen2. **Note:** You cannot use shared key authentication in a streaming mapping.
Client ID	Applicable for Service Principal Authentication. The ID of your application to complete the OAuth Authentication in the Azure Active Directory (AD).
Client Secret	Applicable for Service Principal Authentication. The client secret key to complete the OAuth Authentication in the Azure Active Directory.
Tenant ID	Applicable for Service Principal Authentication. The Directory ID of the Azure Active Directory.
Account Key	Applicable for Shared Key Authentication. The account key for the Microsoft Azure Data Lake Storage Gen2 account.
File System Name	The name of an existing file system in the Microsoft Azure Data Lake Storage Gen2.
Directory Path	The path of an existing directory without the file system name. There is no default directory. You can select one of the following syntax: - `/` for root directory. - `/dir1` - `dir1/dir2`
Adls Gen2 End-point	Type of Microsoft Azure endpoints. You can select any of the following endpoints: - `core.windows.net`: Default - `core.usgovcloudapi.net`: To select the Azure Government endpoints

For more information about creating a client ID, client secret, tenant ID, and file system name, contact the Azure administrator or see Microsoft Azure Data Lake Storage Gen2 documentation.

Microsoft Azure SQL Data Warehouse Connection Properties

Use a Microsoft Azure SQL Data Warehouse connection to access a Microsoft Azure SQL Data Warehouse.

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 SQL Data Warehouse connection in the Administrator tool or the Developer tool. The following table describes the Microsoft Azure SQL Data Warehouse 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 Azure SQL Data Warehouse.

The following table describes the properties for metadata access:

Property	Description
Azure DW JDBC URL	Microsoft Azure Data Warehouse JDBC connection string. For example, you can enter the following connection string: `jdbc:sqlserver:// <Server>.database.windows.net:1433;database=<Database>` The Administrator can download the URL from Microsoft Azure portal.
Azure DW JDBC Username	User name to connect to the Microsoft Azure SQL Data Warehouse account. You must have permission to read, write, and truncate data in Microsoft Azure SQL Data Warehouse.
Azure DW JDBC Password	Password to connect to the Microsoft Azure SQL Data Warehouse account.
Azure DW Schema Name	Name of the schema in Microsoft Azure SQL Data Warehouse.
Azure Storage Type	Type of Azure storage to stage the files. You can select any of the following storage type: - Azure Blob. Default. To use Microsoft Azure Blob Storage to stage the files. - ADLS Gen2. To use Microsoft Azure Data Lake Storage Gen2 as storage to stage the files.
Azure Blob Account Name	Name of the Microsoft Azure Storage account to stage the files.
Azure Blob Account Key	The key that authenticates the access to the Blob storage account.
ADLS Gen2 Storage Account Name	Name of the Microsoft Azure Data Lake Storage Gen2 account to stage the files.
ADLS Gen2 Account Key	Microsoft Azure Data Lake Storage Gen2 access key to stage the files.

Property	Description
Blob End-point	Type of Microsoft Azure endpoints. Select one of the following options: - `core.windows.net`: Default - `core.usgovcloudapi.net`: Select to access the US government Microsoft Azure Data Warehouse endpoints. - `core.chinacloudapi.cn`: Select to access a Microsoft Azure Data Warehouse endpoint in the China region. You can configure the US government Microsoft Azure end-points when a mapping runs in the native environment and on the Spark engine.
VNet Rule	Enable to connect to a Microsoft Azure SQL Data Warehouse endpoint residing in a virtual network (VNet).

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.

Property	Description
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 Connection to Access Sources or Targets

Create connections before you import data objects, preview data, and profile data.

1. Within the Administrator tool click **Manage** > **Connections**.
2. Select **Actions** > **New** > **Connection**.
3. Select the type of connection that you want to create:
   - To select an HBase connection, select **NoSQL** > **HBase**.
   - To select an HDFS connection, select **File Systems** > **Hadoop File System**.
   - To select a Hive connection, select **Database** > **Hive**.
   - To select a JDBC connection, select **Database** > **JDBC**.
4. Click **OK**.
5. Enter a connection name, ID, and optional description.
6. Configure the connection properties. For a Hive connection, you must choose the **Access Hive as a source or target** option to use Hive as a source or a target.
7. Click **Test Connection** to verify the connection.
8. Click **Finish**.

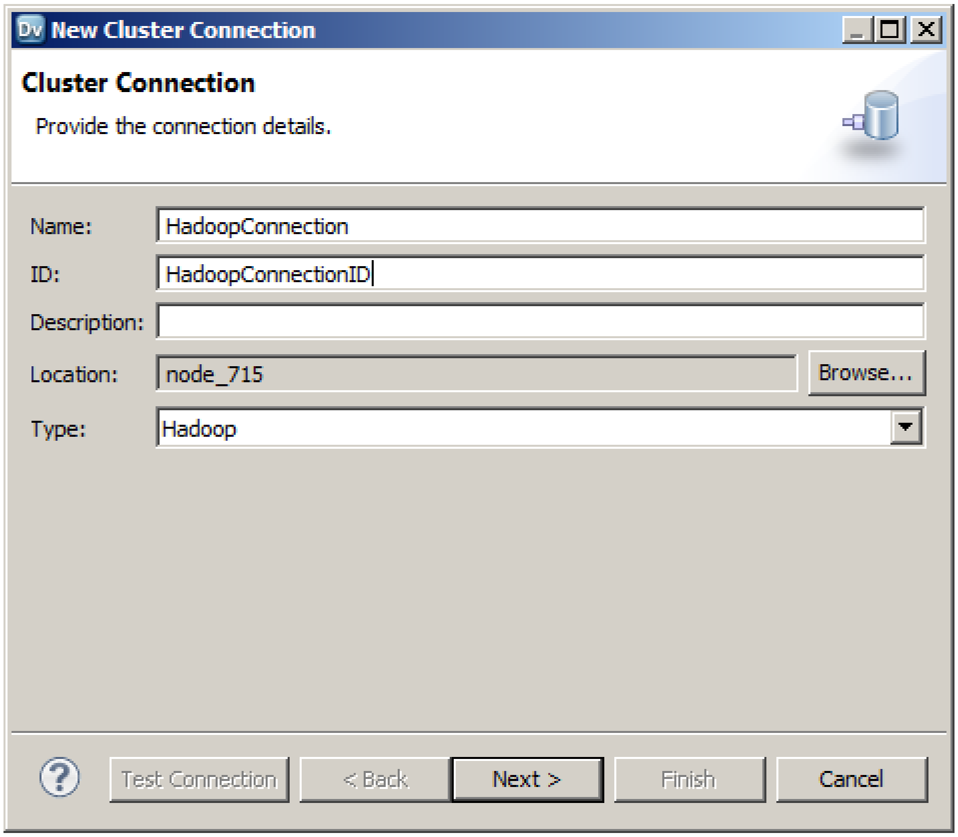
Creating a Hadoop Connection

Create a Hadoop connection before you run a mapping in the Hadoop environment.

1. Click **Window** > **Preferences**.
2. Select **Informatica** > **Connections**.

3. Expand the domain in the **Available Connections** list.
4. Select the **Cluster** connection type in the **Available Connections** list and click **Add**.

   The **New Cluster Connection** dialog box appears.
5. Enter the general properties for the connection.



New Cluster Connection

Cluster Connection

Provide the connection details.

Name: HadoopConnection

ID: HadoopConnectionID

Description:

Location: node_715

Type: Hadoop

6. Click **Next**.
7. Enter the Hadoop cluster properties, common properties, and the reject directory properties.
8. Click **Next**.
9. Click **Next**.

   Effective in version 10.2.2, Informatica dropped support for the Hive engine. Do not enter Hive configuration properties.
10. Enter configuration properties for the Blaze engine and click **Next**.
11. Enter configuration properties for the Spark engine and click **Finish**.

Configuring Hadoop Connection Properties

When you create a Hadoop connection, default values are assigned to cluster environment variables, cluster path properties, and advanced properties. You can add or edit values for these properties. You can also reset to default values.

You can configure the following Hadoop connection properties based on the cluster environment and functionality that you use:

- Cluster Environment Variables
- Cluster Library Path
- Common Advanced Properties
- Blaze Engine Advanced Properties
- Spark Engine Advanced Properties

Note: Informatica does not recommend changing these property values before you consult with third-party documentation, Informatica documentation, or Informatica Global Customer Support. If you change a value without knowledge of the property, you might experience performance degradation or other unexpected results.

To reset to default values, delete the property values. For example, if you delete the values of an edited Cluster Library Path property, the value resets to the default $DEFAULT_CLUSTER_LIBRARY_PATH.

Cluster Environment Variables

Cluster Environment Variables property lists the environment variables that the cluster uses. Each environment variable contains a name and a value. You can add environment variables or edit environment variables.

To edit the property in the text box, use the following format with &: to separate each name-value pair:

```
<name1>=<value1>[&:<name2>=<value2>…&:<nameN>=<valueN>]
```

Configure the following environment variables in the **Cluster Environment Variables** property:

HADOOP_NODE_JDK_HOME

Represents the directory from which you run the cluster services and the JDK version that the cluster nodes use. Required to run the Java transformation in the Hadoop environment and Sqoop mappings on the Blaze engine. Default is /usr/java/default. The JDK version that the Data Integration Service uses must be compatible with the JDK version on the cluster.

Set to <cluster JDK home>/jdk<version>.

For example, `HADOOP_NODE_JDK_HOME=<cluster JDK home>/jdk<version>`.

Cluster Library Path

Cluster Library Path property is a list of path variables for shared libraries on the cluster. You can add or edit library path variables.

To edit the property in the text box, use the following format with : to separate each path variable:

```
<variable1>[:<variable2>…:<variableN]
```

Configure the library path variables in the **Cluster Library Path** property.

Common Advanced Properties

Common advanced properties are a list of advanced or custom properties that are unique to the Hadoop environment. The properties are common to the Blaze and Spark engines. Each property contains a name and a value. You can add or edit advanced properties.

To edit the property in the text box, use the following format with &: to separate each name-value pair:

```
<name1>=<value1>[&:<name2>=<value2>…&:<nameN>=<valueN>]
```

Configure the following property in the **Advanced Properties** of the common properties section:

infapdo.java.opts

List of Java options to customize the Java run-time environment. The property contains default values.

If mappings in a MapR environment contain a Consolidation transformation or a Match transformation, change the following value:

- -Xmx512M. Specifies the maximum size for the Java virtual memory. Default is 512 MB. Increase the value to at least 700 MB.
  For example, `infapdo.java.opts=-Xmx700M`

Blaze Engine Advanced Properties

Blaze advanced properties are a list of advanced or custom properties that are unique to the Blaze engine. Each property contains a name and a value. You can add or edit advanced properties.

To edit the property in the text box, use the following format with &: to separate each name-value pair:

```
<name1>=<value1>[&:<name2>=<value2>…&:<nameN>=<valueN>]
```

Configure the following properties in the **Advanced Properties** of the Blaze configuration section:

infagrid.cadi.namespace

Namespace for the Data Integration Service to use. Required to set up multiple Blaze instances.

Set to <unique namespace>.

For example, `infagrid.cadi.namespace=TestUser1_namespace`

infagrid.blaze.console.jsfport

JSF port for the Blaze engine console. Use a port number that no other cluster processes use. Required to set up multiple Blaze instances.

Set to <unique JSF port value>.

For example, `infagrid.blaze.console.jsfport=9090`

infagrid.blaze.console.httpport

HTTP port for the Blaze engine console. Use a port number that no other cluster processes use. Required to set up multiple Blaze instances.

Set to <unique HTTP port value>.

For example, `infagrid.blaze.console.httpport=9091`

infagrid.node.local.root.log.dir

Path for the Blaze service logs. Default is /tmp/infa/logs/blaze. Required to set up multiple Blaze instances.

Verify that all blaze users have write permission on /tmp.

Set to <local Blaze services log directory>.

For example, `infagrid.node.local.root.log.dir=<directory path>`

infacal.hadoop.logs.directory

Path in HDFS for the persistent Blaze logs. Default is /var/log/hadoop-yarn/apps/informatica. Required to set up multiple Blaze instances.

Set to <persistent log directory path>.

For example, `infacal.hadoop.logs.directory=<directory path>`

infagrid.node.hadoop.local.root.log.dir

Path in the Hadoop connection for the service log directory.

Set to <service log directory path>.

For example, `infagrid.node.local.root.log.dir=$HADOOP_NODE_INFA_HOME/blazeLogs`

Spark Advanced Properties

Spark advanced properties are a list of advanced or custom properties that are unique to the Spark engine. Each property contains a name and a value. You can add or edit advanced properties. Each property contains a name and a value. You can add or edit advanced properties.

Configure the following properties in the **Advanced Properties** of the Spark configuration section:

To edit the property in the text box, use the following format with &: to separate each name-value pair:

```
<name1>=<value1>[&:<name2>=<value2>…&:<nameN>=<valueN>]
```

infasjs.env.spark.context-settings.passthrough.spark.dynamicAllocation.executorIdleTimeout

Maximum time that an Spark Jobserver executor node can be idle before it is removed. Increase the value to assist in debugging data preview jobs that use the Spark engine.

You can specify the time in seconds, minutes, or hours using the suffix *s*, *m*, or *h*, respectively. If you do not specify a time unit, the property uses milliseconds.

If you disable dynamic resource allocation, this property is not used.

Default is 120s.

infasjs.env.spark.jobserver.max-jobs-per-context

Maximum number of Spark jobs that can run simultaneously on a Spark context. If you increase the value of this property, you might need to allocate more resources by increasing *spark.executor.cores* and *spark.executor.memory*.

Default is 10.

infasjs.env.spark.jobserver.sparkJobTimeoutInMinutes

Maximum time in minutes that a Spark job can run on a Spark context before the Spark Jobserver cancels the job. Increase the value to assist in debugging data preview jobs that use the Spark engine.

Default is 15.

infaspark.class.log.level.map

Logging level for specific classes in the Spark driver or executor. When you configure this property, it overrides the tracing level you set for the mapping.

Set the value of this property to a JSON string in the following format: {"<fully qualified class name":"<log level>"}

Join multiple class logging level statements with a comma. You can use the following logging levels: FATAL, WARN, INFO, DEBUG, ALL.

For example, set to:

```
infaspark.class.log.level.map={"org.apache.spark.deploy.yarn.ApplicationMaster":"TRAC
E","org.apache.spark.deploy.security.HadoopFSDelegationTokenProvider":"DEBUG"}
```

infaspark.driver.cluster.mode.extraJavaOptions

List of extra Java options for the Spark driver that runs inside the cluster. Required for streaming mappings to read from or write to a Kafka cluster that uses Kerberos authentication.

For example, set to:

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

To configure the property for a specific user, you can include the following lines of code:

```
infaspark.driver.cluster.mode.extraJavaOptions =
-Djava.security.egd=file:/dev/./urandom
-XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500
-Djava.security.krb5.conf=/etc/krb5.conf
```

infaspark.driver.log.level

Logging level for the Spark driver logs. When you configure this property, it overrides the tracing level you set for the mapping.

Set the value to one of the following levels: FATAL, WARN, INFO, DEBUG, ALL.

infaspark.executor.extraJavaOptions

List of extra Java options for the Spark executor. Required for streaming mappings to read from or write to a Kafka cluster that uses Kerberos authentication.

For example, set to:

```
infaspark.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
```

To configure the property for a specific user, you can include the following lines of code:

```
infaspark.executor.extraJavaOptions =
-Djava.security.egd=file:/dev/./urandom
-XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500
-Djava.security.krb5.conf=/etc/krb5.conf
```

infaspark.executor.log.level

Logging level for the Spark executor logs. When you configure this property, it overrides the tracing level you set for the mapping.

Set the value to one of the following levels: FATAL, WARN, INFO, DEBUG, ALL.

infaspark.flatfile.writer.nullValue

When the Databricks Spark engine writes to a target, it converts null values to empty strings (" "). For example, 12, AB,"",23p09udj.

The Databricks Spark engine can write the empty strings to string columns, but when it tries to write an empty string to a non-string column, the mapping fails with a type mismatch.

To allow the Databricks Spark engine to convert the empty strings back to null values and write to the target, configure the property in the Databricks Spark connection.

Set to: TRUE

infaspark.json.parser.mode

Specifies the parser how to handle corrupt JSON records. You can set the value to one of the following modes:

- DROPMALFORMED. The parser ignores all corrupted records. Default mode.
- PERMISSIVE. The parser accepts non-standard fields as nulls in corrupted records.
- FAILFAST. The parser generates an exception when it encounters a corrupted record and the Spark application goes down.

infaspark.json.parser.multiLine

Specifies whether the parser can read a multiline record in a JSON file. You can set the value to true or false. Default is false. Applies only to non-native distributions that use Spark version 2.2.x and above.

infaspark.pythontx.exec

Required to run a Python transformation on the Spark engine for Data Engineering Integration. The location of the Python executable binary on the worker nodes in the Hadoop cluster.

For example, set to:

```
infaspark.pythontx.exec=/usr/bin/python3.4
```

If you use the installation of Python on the Data Integration Service machine, set the value to the Python executable binary in the Informatica installation directory on the Data Integration Service machine.

For example, set to:

```
infaspark.pythontx.exec=INFA_HOME/services/shared/spark/python/lib/python3.4
```

infaspark.pythontx.executorEnv.LD_PRELOAD

Required to run a Python transformation on the Spark engine for Data Engineering Streaming. The location of the Python shared library in the Python installation folder on the Data Integration Service machine.

For example, set to:

```
infaspark.pythontx.executorEnv.LD_PRELOAD=
INFA_HOME/services/shared/spark/python/lib/libpython3.6m.so
```

infaspark.pythontx.executorEnv.PYTHONHOME

Required to run a Python transformation on the Spark engine for Data Engineering Integration and Data Engineering Streaming. The location of the Python installation directory on the worker nodes in the Hadoop cluster.

For example, set to:

```
infaspark.pythontx.executorEnv.PYTHONHOME=/usr
```

If you use the installation of Python on the Data Integration Service machine, use the location of the Python installation directory on the Data Integration Service machine.

For example, set to:

```
infaspark.pythontx.executorEnv.PYTHONHOME=
INFA_HOME/services/shared/spark/python/
```

infaspark.pythontx.submit.lib.JEP_HOME

Required to run a Python transformation on the Spark engine for Data Engineering Streaming. The location of the Jep package in the Python installation folder on the Data Integration Service machine.

For example, set to:

```
infaspark.pythontx.submit.lib.JEP_HOME=
INFA_HOME/services/shared/spark/python/lib/python3.6/site-packages/jep/
```

infaspark.useHiveWarehouseAPI

Enables the Hive Warehouse Connector. Set to TRUE.

For example, `infaspark.useHiveWarehouseAPI=true`.

spark.authenticate

Enables authentication for the Spark service on Hadoop. Required for Spark encryption.

Set to TRUE.

For example, `spark.authenticate=TRUE`

spark.authenticate.enableSaslEncryption

Enables encrypted communication when SASL authentication is enabled. Required if Spark encryption uses SASL authentication.

Set to TRUE.

For example, `spark.authenticate.enableSaslEncryption=TRUE`

spark.datasource.hive.warehouse.load.staging.dir

Directory for the temporary HDFS files used for batch writes to Hive. Required when you enable the Hive Warehouse Connector.

For example, set to `/tmp`

spark.datasource.hive.warehouse.metastoreUri

URI for the Hive metastore. Required when you enable the Hive Warehouse Connector. Use the value for *hive.metastore.uris* from the hive_site_xml cluster configuration properties.

For example, set the value to `thrift://mycluster-1.com:9083` .

spark.driver.cores

Indicates the number of cores that each driver uses uses to run jobs on the Spark engine.

Set to: `spark.driver.cores=1`

spark.driver.extraJavaOptions

List of extra Java options for the Spark driver.

When you write date/time data within a complex data type to a Hive target using a Hortonworks HDP 3.1 cluster, append the following value to the property: `-Duser.timezone=UTC`

spark.driver.memory

Indicates the amount of driver process memory that the Spark engine uses to run jobs.

Recommended value: Allocate at least 256 MB for every data source.

Set to: `spark.driver.memory=3G`

spark.executor.cores

Indicates the number of cores that each executor process uses to run tasklets on the Spark engine.

Set to: `spark.executor.cores=1`

spark.executor.extraJavaOptions

List of extra Java options for the Spark executor.

When you write date/time data within a complex data type to a Hive target using a Hortonworks HDP 3.1 cluster, append the following value to the property: `-Duser.timezone=UTC`

spark.executor.instances

Indicates the number of instances that each executor process uses to run tasklets on the Spark engine.

Set to: `spark.executor.instances=1`

spark.executor.memory

Indicates the amount of memory that each executor process uses to run tasklets on the Spark engine.

Set to: `spark.executor.memory=3G`

spark.hadoop.hive.llap.daemon.service.hosts

Application name for the LLAP service. Required when you enable the Hive Warehouse Connector. Use the value for *hive.llap.daemon.service.hosts* from the hive_site_xml cluster configuration properties.

spark.hadoop.hive.zookeeper.quorum

Zookeeper hosts used by Hive LLAP. Required when you enable the Hive Warehouse Connector. Use the value for *hive.zookeeper.quorum* from the hive_site_xml cluster configuration properties.

spark.hadoop.validateOutputSpecs

Validates if the HBase table exists. Required for streaming mappings to write to a HBase target in an Amazon EMR cluster. Set the value to false.

spark.scheduler.maxRegisteredResourcesWaitingTime

The number of milliseconds to wait for resources to register before scheduling a task. Default is 30000. Decrease the value to reduce delays before starting the Spark job execution. Required to improve performance for mappings on the Spark engine.

Set to 15000.

For example, `spark.scheduler.maxRegisteredResourcesWaitingTime=15000`

spark.scheduler.minRegisteredResourcesRatio

The minimum ratio of registered resources to acquire before task scheduling begins. Default is 0.8. Decrease the value to reduce any delay before starting the Spark job execution. Required to improve performance for mappings on the Spark engine.

Set to: 0.5

For example, `spark.scheduler.minRegisteredResourcesRatio=0.5`

spark.shuffle.encryption.enabled

Enables encrypted communication when authentication is enabled. Required for Spark encryption.

Set to TRUE.

For example, `spark.shuffle.encryption.enabled=TRUE`

spark.sql.hive.hiveserver2.jdbc.url

URL for HiveServer2 Interactive. Required to use the Hive Warehouse Connector. Use the value in Ambari for HiveServer2 JDBC URL.

spark.yarn.access.hadoopFileSystems

Comma-separated list of external file systems that the Spark service can access. By default, the Spark service has access to the file systems listed in fs.defaultFS in the core-site.xml configuration set of the cluster configuration. Set this property to give the Spark service access to additional file systems.

If you run a mapping on a Cloudera CDP Public Cloud cluster and you use an HDFS on a Cloudera Data Lake cluster, you must allow access to that file system. Append the value for the property *fs.defaultFS* found in `core-site.xml` on the Data Lake cluster. For example:

```
spark.yarn.access.hadoopFileSystems=hdfs://infarndcdppamdl-master1.infarndc.src9-
ltfl.cloudera.site:8020
```

APPENDIX B

Monitoring REST API

This appendix includes the following topics:

Monitoring REST API Overview

You can monitor statistics retrieved through the REST API.

With the REST API, you can get the mapping execution statistics for mappings that run on the native, Blaze, or Spark engines. You can get mapping statistics, mapping advanced statistics, mapping execution steps, and mapping execution plans.

Note: Before you use the REST APIs, verify the following requirements:

- Identify the client or the programming language that you want to use to call the REST APIs.
- Ensure that you start the REST Operations Hub Service to use the REST APIs.
- Configure the monitoring Model Repository Service to get data from the REST APIs.

Enter the mapping jobID as input to the REST API to get the mapping execution statistics. Monitoring REST API requires user name, password encrypted with the pmpasswd command line program, and security domain to connect to the Informatica domain to get the monitoring execution statistics. You can pass the Informatica domain user name, password, and security domain as headers during the REST API.

You can use the following REST APIs to get the following mapping execution statistics:

- MappingStats
- MappingAdvancedStats
- MappingExecutionSteps
- MappingExecutionPlans

Monitoring Metadata Document

Use the metadata document to get a list of monitoring REST APIs. With the monitoring REST APIs, you can get mapping execution statistics and the input and output parameters for each REST API.

You can use the following URL to view the metadata document:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/$metadata
```

Sample Metadata Document

The following sample metadata document displays the mapping execution information for each REST API:

```
<?xml version='1.0' encoding='UTF-8'?>
<edmx:Edmx Version="4.0" xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx">
    <edmx:DataServices>
        <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="OData.Domain"
Alias="MappingService">
            <EntityType Name="MappingAdvancedStat">
                <Key>
                    <PropertyRef Name="jobId"/>
                </Key>
                <Property Name="jobId" Type="Edm.String"/>
                <Property Name="mappingStat" Type="MappingService.MSummaryStatDetails"/>
                <Property Name="detailedStats" Type="MappingService.MDetailedStats"/>
                <Property Name="summaryStats" Type="MappingService.MSrcTgtStats"/>
            </EntityType>
            <EntityType Name="MappingExecutionPlan">
                <Key>
                    <PropertyRef Name="jobId"/>
                </Key>
                <Property Name="jobId" Type="Edm.String"/>
                <Property Name="scriptList"
Type="Collection(MappingService.ScriptEntryStat)"/>
            </EntityType>
            <EntityType Name="MappingStat">
                <Key>
                    <PropertyRef Name="jobId"/>
                </Key>
                <Property Name="jobId" Type="Edm.String"/>
                <Property Name="mappingDetails"
Type="MappingService.MSummaryStatDetails"/>
            </EntityType>
            <EntityType Name="MappingExecutionStep">
                <Key>
                    <PropertyRef Name="jobId"/>
                    <PropertyRef Name="fetchScripts"/>
                </Key>
                <Property Name="jobId" Type="Edm.String"/>
                <Property Name="fetchScripts" Type="Edm.Boolean"/>
                <Property Name="executionStepStats"
Type="Collection(MappingService.MExecutionStepStat)"/>
            </EntityType>
            <ComplexType Name="MHiveScriptStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="hiveHistoryFile" Type="Edm.String"/>
                <Property Name="hiveSessionFile" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MGroupStat">
                <Property Name="processedBytes" Type="Edm.Int64"/>
                <Property Name="processedRows" Type="Edm.Int64"/>
                <Property Name="bytesThrougput" Type="Edm.Int64"/>
                <Property Name="rowsThroughput" Type="Edm.Int64"/>
                <Property Name="errorRows" Type="Edm.Int64"/>
                <Property Name="errorBytes" Type="Edm.Int64"/>
                <Property Name="groupName" Type="Edm.String"/>
                <Property Name="firstRowTime" Type="Edm.Int64"/>
            </ComplexType>
            <ComplexType Name="MDetailedStats">
                <Property Name="memoryData" Type="Collection(Edm.Int64)"/>
                <Property Name="createdTime" Type="Collection(Edm.Int64)"/>
                <Property Name="cpuData" Type="Collection(Edm.Int64)"/>
                <Property Name="sourceTargetStats"
Type="Collection(MappingService.TxInstanceStat)"/>
                <Property Name="lastPurgeTime" Type="Edm.Int64"/>
            </ComplexType>
            <ComplexType Name="MNameValuePair">
                <Property Name="name" Type="Edm.String"/>
                <Property Name="value" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MExecutionStepStat">
                <Property Name="endTime" Type="Edm.Int64"/>
                <Property Name="startTime" Type="Edm.Int64"/>
                <Property Name="errorCode" Type="Edm.String"/>
                <Property Name="errorMessage" Type="Edm.String"/>
                <Property Name="name" Type="Edm.String"/>
                <Property Name="state" Type="Edm.String"/>
                <Property Name="subExecutionStepStat"
Type="Collection(MappingService.MExecutionStepStat)"/>
            </ComplexType>
            <ComplexType Name="MLocalMappingStepStat"
BaseType="MappingService.MExecutionStepStat"/>
            <ComplexType Name="MGridTaskStepStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="percentageCompletion" Type="Edm.Double"/>
                <Property Name="webPageURL" Type="Edm.String"/>
                <Property Name="incomingDependencies" Type="Edm.String"/>
                <Property Name="outgoingDependencies" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="TxInstanceStat">
                <Property Name="instanceName" Type="Edm.String"/>
                <Property Name="isSource" Type="Edm.Boolean"/>
                <Property Name="bytes" Type="Collection(Edm.Int64)"/>
                <Property Name="rows" Type="Collection(Edm.Int64)"/>
            </ComplexType>
            <ComplexType Name="ScriptEntryStat">
                <Property Name="name" Type="Edm.String"/>
                <Property Name="content" Type="Edm.String"/>
                <Property Name="depends" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MCommandTaskStat"
BaseType="MappingService.MExecutionStepStat"/>
            <ComplexType Name="MSrcTgtStats">
                <Property Name="processStatSummary"
Type="MappingService.MProcessStatSummary"/>
                <Property Name="sourceTxStats"
Type="Collection(MappingService.MSourceTxStat)"/>
                <Property Name="targetTxStats"
Type="Collection(MappingService.MTargetTxStat)"/>
            </ComplexType>
            <ComplexType Name="MTargetTxStat" BaseType="MappingService.AbstractTxStat"/>
            <ComplexType Name="MSparkApplicationStepStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="jobTrackerUrl" Type="Edm.String"/>
                <Property Name="query" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MHadoopTezJobStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="dagProgress" Type="Edm.Double"/>
                <Property Name="hadoopJobPageUrl" Type="Edm.String"/>
                <Property Name="hadoopOfficialTezJobId" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MSegmentStepStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="numOfFailedTasklets" Type="Edm.Int64"/>
                <Property Name="numOfSucceededTasklets" Type="Edm.Int64"/>
                <Property Name="numOfTasklets" Type="Edm.Int64"/>
                <Property Name="percentageCompletion" Type="Edm.Double"/>
                <Property Name="webPageURL" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MHiveQueryStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="hiveOfficalHiveQueryId" Type="Edm.String"/>
                <Property Name="query" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MProcessStatSummary">
                <Property Name="avgCpuUsage" Type="Edm.Double"/>
                <Property Name="avgMemUsage" Type="Edm.Int64"/>
            </ComplexType>
            <ComplexType Name="AbstractTxStat">
                <Property Name="instanceName" Type="Edm.String"/>
                <Property Name="groupStats"
Type="Collection(MappingService.MGroupStat)"/>
            </ComplexType>
            <ComplexType Name="MSummaryStatDetails">
                <Property Name="status" Type="Edm.String"/>
                <Property Name="mappingName" Type="Edm.String"/>
                <Property Name="applicationName" Type="Edm.String"/>
                <Property Name="serviceName" Type="Edm.String"/>
                <Property Name="logFileName" Type="Edm.String"/>
                <Property Name="startTime" Type="Edm.Int64"/>
                <Property Name="endTime" Type="Edm.Int64"/>
                <Property Name="executorType" Type="Edm.String"/>
                <Property Name="executingNode" Type="Edm.String"/>
                <Property Name="userName" Type="Edm.String"/>
                <Property Name="securityDomain" Type="Edm.String"/>
            </ComplexType>
            <ComplexType Name="MUserDefinedTaskStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="statsProperties"
Type="Collection(MappingService.MNameValuePair)"/>
            </ComplexType>
            <ComplexType Name="MSourceTxStat" BaseType="MappingService.AbstractTxStat"/>
            <ComplexType Name="MHadoopMapReduceJobStat"
BaseType="MappingService.MExecutionStepStat">
                <Property Name="hadoopJobPageUrl" Type="Edm.String"/>
                <Property Name="hadoopOfficialMapReduceJobId" Type="Edm.String"/>
                <Property Name="mapperProgress" Type="Edm.Double"/>
                <Property Name="reducerProgress" Type="Edm.Double"/>
            </ComplexType>
            <EntityContainer Name="MSCONTAINER">
                <EntitySet Name="MappingAdvancedStats"
EntityType="MappingService.MappingAdvancedStat"/>
                <EntitySet Name="MappingExecutionPlans"
EntityType="MappingService.MappingExecutionPlan"/>
                <EntitySet Name="MappingStats" EntityType="MappingService.MappingStat"/>
                <EntitySet Name="MappingExecutionSteps"
EntityType="MappingService.MappingExecutionStep"/>
            </EntityContainer>
        </Schema>
    </edmx:DataServices>
</edmx:Edmx>
```

ClusterStats

With the ClusterStats REST API, you can view the maximum number of cluster nodes used by a mapping for a cluster configuration in a given time duration.

GET Request

To request information from the ClusterStats, use the following URL:

```
<Rest operations hub service host>:<Rest operations hub service port>/Rest operations hub/
services/v1/mapping service/ClusterStats(startTimeInmillis=[value], endTimeInmillis=[value])
```

The following table describes the attributes in the ClusterStats Get URL:

Field	Type	Description
startTimeInmillis	Long	Required. Specifies the start time in milliseconds.
endTimeInmillis	Long	Required. Specifies the end time in milliseconds.

Tip: You can use the local timezone and convert the date and time in milliseconds by using a web utility such as Current Millis. You can also use the mapping run time to get the time in milliseconds.

Get Response

Return information for the ClusterStats for the specified time duration.

The following table describes the ClusterStats attributes present in the body of the response on the native or Hadoop environment:

Field	Type	Description
startTimeInmillis	Long	Specifies the start time in milliseconds.
endTimeInmillis	Long	Specifies the end time in milliseconds.
clusterStatDetails	String	Specifies the maximum number of nodes used by a mapping for a cluster configuration along with mapping ID, cluster configuration ID, and engine type.

Retrieve Cluster Statistics

Use the script to retrieve the details of the cluster statistics on the Spark environment.

You can use the REST API to retrieve information about the cluster statistics with the following request URL:

```
<Rest operations hub service host>:<Rest operations hub service port>/Rest operations
hub/services/v1/mapping service/
ClusterStats(startTimeInmillis=1554769512503, endTimeInmillis=1564769512503)
```

Cluster Statistics Output

```
"startTimeInmillis": 1554769512503,

"endTimeInmillis": 1564769512503,

"clusterStatDetails": [

"CCOID = ivlhdp755 , EXECUTOR TYPE = SPARK : MAXIMUM NODES USED = 4 , MAPPINGID =
zlxX8LUgEem_6GPOtA8lwg",

"CCOID = invishwas , EXECUTOR TYPE = SPARK : MAXIMUM NODES USED = 2 , MAPPINGID =
UPgZ6LU9EemihPpIGP8fCw",

"CCOID = cdhpr , EXECUTOR TYPE = SPARK : MAXIMUM NODES USED = 1 , MAPPINGID = Yk-
rSLUgEem_6GPOtA8lwg"]
```

MappingStats

With the MappingStats REST API, you can view basic mapping statistics, such as the name and time taken for the job for mappings that run in the native or Hadoop environment.

GET Request

To request information from the MappingStats, use the following URL:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingStats('jobId')
```

The following table describes the attributes in the MappingStats Get URL:

Field	Type	Description
userName	String	Required. User name to connect to the domain. You can pass the input value as a header.
encryptedpassword	String	Required. Password for the user. Encrypt the password with the pmpasswd command line program. You can pass the input value as a header.
securityDomain	String	Optional. The security domain to which the domain user belongs. You can pass the input value as a header. Default is Native.
jobId	String	Required. The argument of the entity that contains the ID for the mappings. You can pass the property as part of the URL.

Get Response

Return information for the MappingStats for the specified Job ID.

The following table describes the MappingStats attributes present in the body of the response on the native or Hadoop environment:

Field	Type	Description
jobId	String	The argument of the entity that contains the ID for the mapping.
mappingDetails	n/a	Container for mapping run information, such as mapping name, start time, and end time.
status	String	State of the job run.
mappingName	String	Name of the mapping.
applicationName	String	Name of the application.
serviceName	String	Name of the service.
logFileName	String	Location of the log file.
startTime	Integer	Start time of the job.

Field	Type	Description
endTime	Integer	End time of the job.
executorType	String	Type of the run-time environment where you run the mapping.
executingNode	String	The Data Integration Service node where the job ran.
userName	String	User who started the job.
securityDomain	String	The security domain to which the domain user belongs.

Sample Retrieve Mapping Statistics

The sample use case is to use the script to retrieve the details of the mapping statistics on the Spark environment.

You can use the REST API to retrieve information about the mapping statistics with the following request URL for a mapping with Job ID as _TNoO9ELEeiimY76kFyfuw:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingStats('_TNoO9ELEeiimY76kFyfuw')
```

Mapping Statistics Output

```
{
    "@odata.context": "$metadata#MappingStats/$entity",
    "jobId": "_TNoO9ELEeiimY76kFyfuw",
    "mappingDetails": {
        "status": "COMPLETED",
        "mappingName": "HDFSTgtAppend_MultiPartition_SparkMode",
        "applicationName": "HDFSTargetAppend",
        "serviceName": "DIS_HDP_2.6",
        "logFileName": "/data/Informatica/10.2.2_252/logs/node252/services/
DataIntegrationService/disLogs/ms/
DEPLOYED_MAPPING_HDFSTargetAppend_HDFSTgtAppend_MultiPartition_SparkMode-
_TNoO9ELEeiimY76kFyfuw_20181016_115325_006.log",
        "startTime": 1539671005830,
        "endTime": 1539671244752,
        "executorType": "SPARK",
        "executingNode": "node252",
        "userName": "Administrator",
        "securityDomain": "Native"
    }
}
```

MappingAdvancedStats

With the MappingAdvancedStats REST API, you can view mapping summary and detailed statistics, such as the job throughput for mappings that run in the native or Hadoop environment.

GET Request

To request information from the MappingAdvancedStats, use the following URL:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingAdvancedStats('jobId')
```

The following table describes the attributes in the MappingAdvancedStats Get URL:

Field	Type	Description
userName	String	Required. User name to connect to the domain. You can pass the input value as a header.
encryptedpassword	String	Required. Password for the user. Encrypt the password with the pmpasswd command line program. You can pass the input value as a header.
securityDomain	String	Optional. The security domain to which the domain user belongs. You can pass the input value as a header. Default is Native.
jobId	String	Required. The argument of the entity that contains the ID for the mappings. You can pass the property as part of the URL.

Get Response

Return information for the MappingAdvancedStats for the specified Job ID.

The following table describes the MappingAdvancedStats attributes present in the body of the response for native or Hadoop environment:

Field	Type	Description
jobId	String	The argument of the entity that contains the ID for the mapping.
mappingStat	n/a	Container for mapping run information, such as mapping name, start time, and end time.
status	String	State of the job run.
mappingName	String	Name of the mapping.
applicationName	String	Name of the application.
serviceName	String	Name of the service.
logFileName	String	Location of the log file.
startTime	Integer	Start time of the job.
endTime	Integer	End time of the job.
executorType	String	Type of the run-time environment where you run the mapping.
executingNode	String	The Data Integration Service node where the job ran.
userName	String	User who started the job.
securityDomain	String	The security domain to which the domain user belongs.
detailedStats	n/a	Container for detailed mapping statistics.

Field	Type	Description
memoryData	String	Metrics for memory.
createdTime	Integer	Time taken to created the job.
cpuData	Integer	Data for CPU.
sourceTargetStats	n/a	Container for source and target statistics.
instanceName	String	Object name for source or target.
isSource	Integer	For the source, specify isSource as true. For the target, specify isSource as false.
bytes	Integer	Average number of bytes read per second for source or target.
rows	Integer	Number of rows read for source or target.
lastPurgeTime	Integer	Last time of the purge task.
summaryStats	n/a	Container for the summary statistics.
processStatSummary	n/a	Container for the CPU and memory usage.
avgCpuUsage	Integer	Average CPU usage.
avgMemUsage	Integer	Average memory usage.
sourceTxStats	n/a	Container for the source statistics.
groupStats	n/a	Container for processed and throughput statistics.
ProcessedBytes	Integer	Bytes processed for source or target.
ProcessedRows	Integer	Rows processed for source or target.
bytesThrougput	Integer	Throughput in bytes for source or target.
rowsThroughput	Integer	Throughput of rows for source or target.
sourceThroughput	Integer	Mean rate at which messages are ingested to source.
targetThroughput	Integer	Mean rate at which messages are written to target.
errorRows	Integer	Error in rows for source or target.
errorBytes	Integer	Error in bytes for source or target.
groupName	String	Name of the group for source or target.
firstRowTime	Integer	Time taken for the first row for source or target.
targetTxStats	n/a	Container for target statistics.

Sample Retrieve Advanced Mapping Statistics

The sample use case is to use the script to retrieve the details of the advanced mapping statistics on the Spark environment.

You can use the REST API to retrieve information about the advanced mapping statistics with the following request URL for a mapping with Job ID as _TNoO9ELEeiimY76kFyfuw:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingAdvancedStats('_TNoO9ELEeiimY76kFyfuw')
```

Advanced Mapping Statistics Output

```
{
    "@odata.context": "$metadata#MappingAdvancedStats/$entity",
    "jobId": "_TNoO9ELEeiimY76kFyfuw",
    "mappingStat": {
        "status": "COMPLETED",
        "mappingName": "HDFSTgtAppend_MultiPartition_SparkMode",
        "applicationName": "HDFSTargetAppend",
        "serviceName": "DIS_HDP_2.6",
        "logFileName": "/data/Informatica/10.2.2_252/logs/node252/services/
DataIntegrationService/disLogs/ms/
DEPLOYED_MAPPING_HDFSTargetAppend_HDFSTgtAppend_MultiPartition_SparkMode-
TNoO9ELEeiimY76kFyfuw_20181016_115325_006.log",
        "startTime": 1539671005830,
        "endTime": 1539671244752,
        "executorType": "SPARK",
        "executingNode": "node252",
        "userName": "Administrator",
        "securityDomain": "Native"
    },
    "detailedStats": {
        "memoryData": [],
        "createdTime": [
            1539671058384
        ],
        "cpuData": [],
        "sourceTargetStats": [
            {
                "instanceName": "Read_students_5",
                "isSource": true,
                "bytes": [
                    -1
                ],
                "rows": [
                    10
                ]
            },
            {
                "instanceName": "Read_students_HDFS_src",
                "isSource": true,
                "bytes": [
                    -1
                ],
                "rows": [
                    10
                ]
            },
            {
                "instanceName": "Read_student",
                "isSource": true,
                "bytes": [
                    -1
                ],
                "rows": [
                    10
                ]
            },
            {
                "instanceName": "Write_HDFSAppendTarget",
                "isSource": false,
                "bytes": [
                    -1
                ],
                "rows": [
                    28
                ]
            }
        ],
        "lastPurgeTime": 0
    },
    "summaryStats": {
        "processStatSummary": {
            "avgCpuUsage": 0,
            "avgMemUsage": 0
        },
        "sourceTxStats": [
            {
                "instanceName": "Read_students_5",
                "groupStats": [
                    {
                        "processedBytes": -1,
                        "processedRows": 10,
                        "bytesThrougput": -1,
                        "rowsThroughput": 10,
                        "errorRows": -1,
                        "errorBytes": -1,
                        "groupName": "Read_students_5",
                        "firstRowTime": 0
                    }
                ]
            },
            {
                "instanceName": "Read_student",
                "groupStats": [
                    {
                        "processedBytes": -1,
                        "processedRows": 10,
                        "bytesThrougput": -1,
                        "rowsThroughput": 10,
                        "errorRows": -1,
                        "errorBytes": -1,
                        "groupName": "Read_student",
                        "firstRowTime": 0
                    }
                ]
            },
            {
                "instanceName": "Read_students_HDFS_src",
                "groupStats": [
                    {
                        "processedBytes": -1,
                        "processedRows": 10,
                        "bytesThrougput": -1,
                        "rowsThroughput": 5,
                        "errorRows": -1,
                        "errorBytes": -1,
                        "groupName": "Read_students_HDFS_src",
                        "firstRowTime": 0
                    }
                ]
            }
        ],
        "targetTxStats": [
            {
                "instanceName": "Write_HDFSAppendTarget",
                "groupStats": [
                    {
                        "processedBytes": -1,
                        "processedRows": 28,
                        "bytesThrougput": -1,
                        "rowsThroughput": 14,
                        "errorRows": -1,
                        "errorBytes": -1,
                        "groupName": "Write_HDFSAppendTarget",
                        "firstRowTime": 0
                    }
                ]
            }
        ]
    }
}
```

MappingExecutionSteps

With the MappingExecutionStats REST API, you can view execution steps for Hadoop jobs. You can also view the scripts for Hadoop jobs with the fetchScripts option.

GET Request

To request information from the MappingExecutionSteps with the associated scripts within the Hadoop scripts, use the following URL:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingExecutionSteps(jobId='id',fetchScripts=true)
```

To request information from the MappingExecutionSteps without the associated scripts within the Hadoop scripts, use the following URL:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingExecutionSteps(jobId='id',fetchScripts=false)
```

The following table describes the required attributes in the MappingExecutionSteps Get URL:

Field	Type	Description
userName	String	Required. User name to connect to the domain. You can pass the input value as a header.
encryptedpassword	String	Required. Password for the user. Encrypt the password with the pmpasswd command line program. You can pass the input value as a header.
securityDomain	String	Optional. The security domain to which the domain user belongs. You can pass the input value as a header.
jobId	String	Required. The argument of the entity that contains the ID for the mappings. You can pass the input value as a query.
fetchScripts	String	Required. Gets all associated scripts within the Hadoop scripts. You can enter true to get the associated run tasks with the scripts. To get the associated run tasks without the scripts, enter false.

Get Response

Return information for the MappingExecutionSteps for the specified Job ID.

The following table describes the MappingExecutionSteps attributes present in the body of the response for Blaze environment:

Field	Description
Session Task	Response can include three types of session tasks that are pre session task, main session task, and post session task. The main session task can contain Submappings.
Instances	Transformation or object name.
endTime	End time of the job.
startTime	Start time of the job.
errorCode	Error code.
name	Name of the submapping task based on the run-time environment.
state	State of the mapping run.
subExecutionStepStat	Statistics for the sub execution step.
numOfFailedTasklets	Number of failed tasklets.
numOfSucceededTasklets	Number of succeeded tasklets.
numOfTasklets	Number of tasklets.
percentageCompletion	Percentage completion of tasklets.
webPageURL	Blaze monitoring tasklet URL.
incomingDependencies	Incoming dependencies contain pre session tasks.
outgoingDependencies	Outgoing dependencies contain post session tasks.

The following table describes the MappingExecutionSteps attributes present in the body of the response for Spark environment:

Field	Type	Description
jobId	String	The argument of the entity that contains the ID for the mapping.
fetchScripts	String	Gets all associated scripts within the Spark scripts. You can enter true to get the associated run tasks with the scripts. To get the associated run tasks without the scripts, enter false.
executionStepStats	n/a	Container for the statistics related to mapping execution steps.
endTime	Integer	End time of the job.
startTime	Integer	Start time of the job.
errorCode	Integer	Error code.

Field	Type	Description
errorMessage	Integer	Error message.
name	String	Name of the main mapping task based on the run-time environment.
state	String	State of the job run.
subExecutionStepStats	Integer	Statistics for the subexecution step statistics.
jobTrackerURL	String	The URL to the job tracker.
query	String	The main Spark query.

Sample Retrieve Mapping Execution Steps

The sample use case is to use the script to retrieve the details of the mapping execution steps for the Spark environment.

You can use the REST API to retrieve information about the mapping execution steps with the following request URL for a mapping with Job ID as _TNoO9ELEeiimY76kFyfuw:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/
MappingExecutionSteps(jobId='_TNoO9ELEeiimY76kFyfuw',fetchScripts=true)
```

Mapping Execution Steps Output

```
{
    "@odata.context": "$metadata#MappingExecutionSteps/$entity",
    "jobId": "_TNoO9ELEeiimY76kFyfuw",
    "fetchScripts": true,
    "executionStepStats": [
        {
            "@odata.type": "#OData.Domain.MSparkApplicationStepStat",
            "endTime": 1539671244359,
            "startTime": 1539671066452,
            "errorCode": "",
            "errorMessage": "",
            "name": "InfaSpark0",
            "state": "COMPLETED",
            "subExecutionStepStat": [],
            "jobTrackerUrl":
"https://ivlhdp621.informatica.com:8090/proxy/application_1539263790092_1418",
            "query": "package com.informatica.exec\n\nimport
com.informatica.bootstrap.functions._\nimport com.informatica.bootstrap.InfaParams._
\nimport com.informatica.bootstrap.InfaStreaming.writeToKafka\nimport
com.informatica.products.infatransform.spark.boot._\nimport com.informatica.bootstrap._
\nimport com.informatica.hive._\nimport com.informatica.bootstrap.{JsonProtocol => JP}
\nimport org.apache.spark._\nimport org.apache.spark.rdd._\nimport
org.apache.spark.storage.StorageLevel._\nimport org.apache.spark.sql._\nimport
org.apache.spark.sql.types._\nimport org.apache.spark.sql.functions._\nimport
org.apache.spark.sql.functions.{ broadcast => infabroadcast }\nimport
org.apache.spark.sql.infa.expressions._\nimport java.io._\nimport java.sql.Timestamp
\nimport scala.reflect.ClassTag\nimport org.apache.spark.sql.catalyst.expressions.Caster
\nimport org.apache.spark.sql.catalyst.expressions.JavaCaster\nimport
com.informatica.bootstrap.JavaTx._\nimport org.apache.spark.Accumulator\nimport
org.apache.spark.util.LongAccumulator\nimport org.apache.spark.scheduler.SparkListener
\nimport org.apache.spark.SparkEnv\nimport org.apache.spark.sql.Row\n\nobject InfaSpark0
{\n  def main(s:Array[String]) {\n    val sc = SparkContextLoader.getSparkContext\n
val sqlContext = SparkContextLoader.getSQLContext\n    val ls = new
LiveStream(sc.getConf)\n    ls.relay(JP.sparkConfToJson(sc.getConf)) \n
ls.relay(JP.hadoopConfToJson(sc.hadoopConfiguration)) \n    val lis = new
InfaListener(ls,\"TAG\")\n    sc.addSparkListener(lis)
\nsqlContext.sparkSession.experimental.extraPreprocessing = new InfaTaggingRules().rules
\n    val accs = List()\n    ls.relay(JP.sparkAppDetailsToJson(sc.getConf, accs)) \n
lis.accumulators = accs\n    import sqlContext.implicits._\n    import
org.apache.spark.sql.functions.{stddev_samp, var_samp}\n    val icast = caster(\"MM/DD/
YYYY HH24:MI:SS\")\n    val acast = adapterCaster()\n    val jcast = JavaCaster()\n\n
try {\n    Tuple2(sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\",
\"0\"))), sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\",
\"1\"))));\n    Tuple2(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"2\"))), sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"3\"))));\n
Tuple2(sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"4\"))),
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"5\"))));\n    val
v0 = infabroadcast(asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"6\")))).tag(\"SRC_Read_students_5\").itoDF(\"m\")).itoDF;\n    val v1
= updatePartitions(asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"7\")))).tag(\"SRC_Read_students_HDFS_src\").itoDF(\"d\"), v0);\n
val v2 = v1.join(v0, v0(0).===(v1(0)), \"inner\").itoDF(\"m\");\n    val v3 =
updatePartitions(asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"8\")))).tag(\"SRC_Read_student\").itoDF(\"d\"), v2);\n    val v4 =
v3.join(v2, v2(1).===(v3(0)), \"inner\").itoDF;\n    val v5 =
DataTypes.createDecimalType(28, 0);\n    val v6 = DataTypes.createDecimalType(18,
0);\n    asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\",
\"9\"))), v4.iselect(icast(icast(v4(2), v5), v6), v4(3), v4(4), icast(icast(v4(5), v5),
v6)).itoDF(\"TGT_\").tag(\"TGT_Write_HDFSAppendTarget\").itoDF(\"c
\").createOrReplaceTempView(\"tbl0\"));\n    } finally {\n
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"10\")));\n
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"11\")));\n
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"12\")));\n    }
\n    sc.stop\n}\n}\n[0] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode`]\n[1] -> [CREATE TABLE
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode` (`col0` INT, `col1` STRING, `col2` STRING, `col3` INT, `col4` STRING) ROW
FORMAT SERDE 'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFASerDe'
STORED AS INPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAInputFormat'
OUTPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat'
LOCATION 'hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_infa_Read_students_HDFS_src_HDFSTgtAppend_MultiPartition_SparkMode'
TBLPROPERTIES ('infa.columns.types'='int,string,string,int,string',
'pwx.mapping.file.path'='./
Read_students_HDFS_src_MAPPING_37960411407997671_37960411786739094.bin',
'auto.purge'='true', 'infa.columns'='col0,col1,col2,col3,col4')]\n[2] -> [DROP TABLE IF
EXISTS
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de`]\n[3] -> [CREATE TABLE
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de` (`col0` INT, `col1` STRING, `col2` STRING, `col3` INT) ROW FORMAT SERDE
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFASerDe' STORED AS
INPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAInputFormat'
OUTPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat'
LOCATION 'hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_INFACOPY_Read_students_5_HDFSTgtAppend_MultiPartition_SparkMode'
TBLPROPERTIES ('infa.columns.types'='int,string,string,int', 'pwx.mapping.file.path'='./
Read_students_5_MAPPING_37960411392603831_37960411887963169.bin', 'auto.purge'='true',
'infa.columns'='col0,col1,col2,col3')]\n[4] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode`]\n[5] -> [CREATE TABLE
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode` (`col0` DECIMAL(18, 0), `col1` STRING, `col2` STRING, `col3` DECIMAL(18, 0))
ROW FORMAT SERDE
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFASerDe' STORED AS
INPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAInputFormat'
OUTPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat'
LOCATION 'hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_infa_Write_HDFSAppendTarget_HDFSTgtAppend_MultiPartition_SparkMode'
```

```
TBLPROPERTIES ('infa.columns.types'='decimal(18,0),string,string,decimal(18,0)',
'pwx.mapping.file.path'='./
Write_HDFSAppendTarget_MAPPING_37960411526174778_37960411903682194.bin',
'pwx.skip.serialization'='true', 'auto.purge'='true',
'infa.columns'='col0,col1,col2,col3')]\n[6] -> [SELECT
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col0`
 as a0,
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col1`
 as a1,
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col2`
 as a2,
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col3`
 as a3 FROM
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de`]\n[7] -> [SELECT
`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_sparkmode`
.`col0` as a0 FROM
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode`]\n[8] -> [SELECT alias.id as a0 FROM DEFAULT.student alias]\n[9] -> [INSERT
OVERWRITE TABLE
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode` SELECT tbl0.c0 as a0, tbl0.c1 as a1, tbl0.c2 as a2, tbl0.c3 as a3 FROM
tbl0]\n[10] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode`]\n[11] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de`]\n[12] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode`]"
        }
    ]
}
```

MappingExecutionPlans

With the MappingExecutionPlans REST API, you can view the execution plan for Hadoop jobs.

GET Request

To request information from the MappingExecutionPlans, use the following URL:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/RestOperationsHub/
services/v1/MappingService/MappingExecutionPlans('jobId')
```

The following table describes the required attributes in the MappingExecutionPlans Get URL:

Field	Type	Description
userName	String	Required. User name to connect to the domain. You can pass the input value as a header.
encryptedpassword	String	Required. Password for the user. Encrypt the password with the pmpasswd command line program. You can pass the input value as a header.

Field	Type	Description
securityDomain	String	Optional. The security domain to which the domain user belongs. You can pass the input value as a header.
jobId	String	Required. The argument of the entity that contains the ID for the mappings. You can pass the input value as a query.

Get Response

Return information for the MappingExecutionPlans for the specified Job ID.

The following table describes the MappingExecutionPlans attributes present in the body of the response for the Spark or Blaze environment:

Field	Type	Description
name	Integer	Name of the script.
content	String	For Spark, the Data Integration Service translates the mapping to a Scala program and an optional set of commands. The execution plan shows the commands and the Scala program code. For Blaze, the content comprises of session task, instances, and type.
depends	String	Tasks that the script depends on. Tasks include other scripts and Data Integration Service tasks, like the Start task.

The following table describes the MappingExecutionPlans attributes present in the content section of the body of the response for Blaze environment:

Field	Description
Session Task	Response can include three types of session tasks that are pre session task, main session task, and post session task. The main session task can contain Submappings.
Type	Type of the session task containing a set of segments and DAG vertices.
Instances	Transformation or object name.

Sample Retrieve Mapping Execution Plans

The sample use case is to use the script to retrieve the details of the mapping execution plans.

You can use the REST API to retrieve information about the mapping execution plans with the following request URL for a mapping with Job ID as _TNoO9ELEeiimY76kFyfuw:

```
<RESTOperationsHubService_Host>:<RESTOperationsHubService_Port>/restopshub/services/v1/
MappingService/MappingExecutionPlans('_TNoO9ELEeiimY76kFyfuw')
```

Mapping Execution Plans Output

```
{
    "@odata.context": "$metadata#MappingExecutionPlans/$entity",
    "jobId": "_TNoO9ELEeiimY76kFyfuw",
    "scriptList": [
        {
            "name": "InfaSpark0",
            "content": "package com.informatica.exec\n\nimport
com.informatica.bootstrap.functions._\nimport com.informatica.bootstrap.InfaParams._
\nimport com.informatica.bootstrap.InfaStreaming.writeToKafka\nimport
com.informatica.products.infatransform.spark.boot._\nimport com.informatica.bootstrap._
\nimport com.informatica.hive._\nimport com.informatica.bootstrap.{JsonProtocol => JP}
\nimport org.apache.spark._\nimport org.apache.spark.rdd._\nimport
org.apache.spark.storage.StorageLevel._\nimport org.apache.spark.sql._\nimport
org.apache.spark.sql.types._\nimport org.apache.spark.sql.functions._\nimport
org.apache.spark.sql.functions.{ broadcast => infabroadcast }\nimport
org.apache.spark.sql.infa.expressions._\nimport java.io._\nimport java.sql.Timestamp
\nimport scala.reflect.ClassTag\nimport org.apache.spark.sql.catalyst.expressions.Caster
\nimport org.apache.spark.sql.catalyst.expressions.JavaCaster\nimport
com.informatica.bootstrap.JavaTx._\nimport org.apache.spark.Accumulator\nimport
org.apache.spark.util.LongAccumulator\nimport org.apache.spark.scheduler.SparkListener
\nimport org.apache.spark.SparkEnv\nimport org.apache.spark.sql.Row\n\nobject InfaSpark0
{\n  def main(s:Array[String]) {\n    val sc = SparkContextLoader.getSparkContext\n
val sqlContext = SparkContextLoader.getSQLContext\n    val ls = new
LiveStream(sc.getConf)\n    ls.relay(JP.sparkConfToJson(sc.getConf)) \n
ls.relay(JP.hadoopConfToJson(sc.hadoopConfiguration)) \n    val lis = new
InfaListener(ls,\"TAG\")\n    sc.addSparkListener(lis)
\nsqlContext.sparkSession.experimental.extraPreprocessing = new InfaTaggingRules().rules
\n    val accs = List()\n    ls.relay(JP.sparkAppDetailsToJson(sc.getConf, accs)) \n
lis.accumulators = accs\n    import sqlContext.implicits._\n    import
org.apache.spark.sql.functions.{stddev_samp, var_samp}\n    val icast = caster(\"MM/DD/
YYYY HH24:MI:SS\")\n    val acast = adapterCaster()\n    val jcast = JavaCaster()\n\n
try {\n    Tuple2(sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\",
\"0\"))), sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\",
\"1\"))));\n    Tuple2(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"2\"))), sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"3\"))));\n
Tuple2(sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"4\"))),
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"5\"))));\n    val
v0 = infabroadcast(asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"6\")))).tag(\"SRC_Read_students_5\").itoDF(\"m\")).itoDF;\n    val v1
= updatePartitions(asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"7\")))).tag(\"SRC_Read_students_HDFS_src\").itoDF(\"d\"), v0);\n
val v2 = v1.join(v0, v0(0).===(v1(0)), \"inner\").itoDF(\"m\");\n    val v3 =
updatePartitions(asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String]
(\"InfaSpark0\", \"8\")))).tag(\"SRC_Read_student\").itoDF(\"d\"), v2);\n    val v4 =
v3.join(v2, v2(1).===(v3(0)), \"inner\").itoDF;\n    val v5 =
DataTypes.createDecimalType(28, 0);\n    val v6 = DataTypes.createDecimalType(18,
0);\n    asBlock(sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\",
\"9\"))), v4.iselect(icast(icast(v4(2), v5), v6), v4(3), v4(4), icast(icast(v4(5), v5),
v6)).itoDF(\"TGT_\").tag(\"TGT_Write_HDFSAppendTarget\").itoDF(\"c
\").createOrReplaceTempView(\"tbl0\"));\n    } finally {\n
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"10\")));\n
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"11\")));\n
sqlContext.sql(InfaParams.resolve(InfaParams[String](\"InfaSpark0\", \"12\")));\n    }
\n    sc.stop\n}\n}\n[0] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode`]\n[1] -> [CREATE TABLE
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode` (`col0` INT, `col1` STRING, `col2` STRING, `col3` INT, `col4` STRING) ROW
FORMAT SERDE 'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFASerDe'
STORED AS INPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAInputFormat'
OUTPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat'
LOCATION 'hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_infa_Read_students_HDFS_src_HDFSTgtAppend_MultiPartition_SparkMode'
TBLPROPERTIES ('infa.columns.types'='int,string,string,int,string',
'pwx.mapping.file.path'='./
Read_students_HDFS_src_MAPPING_37960411407997671_37960411786739094.bin',
'auto.purge'='true', 'infa.columns'='col0,col1,col2,col3,col4')]\n[2] -> [DROP TABLE IF
EXISTS
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de`]\n[3] -> [CREATE TABLE
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de` (`col0` INT, `col1` STRING, `col2` STRING, `col3` INT) ROW FORMAT SERDE
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFASerDe' STORED AS
INPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAInputFormat'
OUTPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat'
LOCATION 'hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_INFACOPY_Read_students_5_HDFSTgtAppend_MultiPartition_SparkMode'
TBLPROPERTIES ('infa.columns.types'='int,string,string,int', 'pwx.mapping.file.path'='./
Read_students_5_MAPPING_37960411392603831_37960411887963169.bin', 'auto.purge'='true',
'infa.columns'='col0,col1,col2,col3')]\n[4] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode`]\n[5] -> [CREATE TABLE
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode` (`col0` DECIMAL(18, 0), `col1` STRING, `col2` STRING, `col3` DECIMAL(18, 0))
ROW FORMAT SERDE
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFASerDe' STORED AS
INPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAInputFormat'
OUTPUTFORMAT
'com.informatica.platform.dtm.executor.hive.boot.storagehandler.INFAOutputFormat'
LOCATION 'hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_infa_Write_HDFSAppendTarget_HDFSTgtAppend_MultiPartition_SparkMode'
TBLPROPERTIES ('infa.columns.types'='decimal(18,0),string,string,decimal(18,0)',
'pwx.mapping.file.path'='./
Write_HDFSAppendTarget_MAPPING_37960411526174778_37960411903682194.bin',
'pwx.skip.serialization'='true', 'auto.purge'='true',
'infa.columns'='col0,col1,col2,col3')]\n[6] -> [SELECT
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col0`
 as a0,
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col1`
 as a1,
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col2`
 as a2,
`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmode`.`col3`
 as a3 FROM
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de`]\n[7] -> [SELECT
`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_sparkmode`
.`col0` as a0 FROM
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode`]\n[8] -> [SELECT alias.id as a0 FROM DEFAULT.student alias]\n[9] -> [INSERT
OVERWRITE TABLE
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode` SELECT tbl0.c0 as a0, tbl0.c1 as a1, tbl0.c2 as a2, tbl0.c3 as a3 FROM
tbl0]\n[10] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_write_hdfsappendtarget_hdfstgtappend_multipartition_
sparkmode`]\n[11] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_read_students_5_hdfstgtappend_multipartition_sparkmo
de`]\n[12] -> [DROP TABLE IF EXISTS
`default`.`w7939778750618549156_infa_read_students_hdfs_src_hdfstgtappend_multipartition_
sparkmode`]",
            "depends": "Pre_Spark_Task_Command_1"
        },
        {
            "name": "Pre_Spark_Task_Command_1",
            "content": "-- ----\n-- Command [Pre_Spark_Task_Command_1_1] \n/data/
Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6/scripts/FileCopyUtil --
hadoop.home /data/Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6 --hdfsUser
impUser1 --copyFromLocal --spn adpqa@INFAKRB.INFADEV.COM --keytab /bdmqa/BDM_Automation/
Source/adpqa_AD.keytab --ccoConfPath /data/Informatica/10.2.2_252/tomcat/bin/disTemp/
inkrh71hdp07_252/DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-site.xml file:///bdmqa/
BDM_Automation/Source/students_10.txt hdfs://nameservice1//tmp/SPARK_impUser1/
sess7939778750618549156//
W7939778750618549156_INFACOPY_Read_students_5_HDFSTgtAppend_MultiPartition_SparkMode/
students_10.txt\n-- ----\n-- Command [Pre_Spark_Task_Command_1_2] \n/data/Informatica/
10.2.2_252/services/shared/hadoop/HDP_2.6/scripts/HadoopFsMkdir --hadoop.home /data/
Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6 --hdfsUser impUser1 --spn
adpqa@INFAKRB.INFADEV.COM --keytab /bdmqa/BDM_Automation/Source/adpqa_AD.keytab --
ccoConfPath /data/Informatica/10.2.2_252/tomcat/bin/disTemp/inkrh71hdp07_252/
DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-site.xml hdfs://nameservice1//tmp/SPARK_impUser1/
sess7939778750618549156//
W7939778750618549156_Write_HDFSAppendTarget_HDFSTgtAppend_MultiPartition_SparkMode/
```

```
reject-files/\n-- ----\n-- Command [Pre_Spark_Task_Command_1_3] \n/data/Informatica/
10.2.2_252/services/shared/hadoop/HDP_2.6/scripts/HadoopFsRmRf --hadoop.home /data/
Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6 --hdfsUser impUser1 --spn
adpqa@INFAKRB.INFADEV.COM --keytab /bdmqa/BDM_Automation/Source/adpqa_AD.keytab --
ccoConfPath /data/Informatica/10.2.2_252/tomcat/bin/disTemp/inkrh71hdp07_252/
DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-site.xml hdfs://nameservice1/BDM_Automation/Target/
HDFSTargetAppend/SparkMode/MultiPartition//StudentHDFSTargetAppend.out-[mr]?[0-9]*\n--
----\n-- Command [Pre_Spark_Task_Command_1_4] \ntouch /bdmqa/BDM_Automation/Target/
StudentHDFSTargetAppend61c555e1-c967-4888-bc7d-ab9ada8ee2a7_empty.txt\n-- ----\n--
Command [Pre_Spark_Task_Command_1_5] \n/data/Informatica/10.2.2_252/services/shared/
hadoop/HDP_2.6/scripts/FileCopyUtil --hadoop.home /data/Informatica/10.2.2_252/services/
shared/hadoop/HDP_2.6 --hdfsUser impUser1 --copyFromLocal --spn
adpqa@INFAKRB.INFADEV.COM --keytab /bdmqa/BDM_Automation/Source/adpqa_AD.keytab --
ccoConfPath /data/Informatica/10.2.2_252/tomcat/bin/disTemp/inkrh71hdp07_252/
DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-site.xml file:///bdmqa/BDM_Automation/Target/
StudentHDFSTargetAppend61c555e1-c967-4888-bc7d-ab9ada8ee2a7_empty.txt hdfs://
nameservice1/BDM_Automation/Target/HDFSTargetAppend/SparkMode/MultiPartition//
StudentHDFSTargetAppend.out\n-- ----\n-- Command [Pre_Spark_Task_Command_1_6] \n/data/
Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6/scripts/HadoopFsRmRf --
hadoop.home /data/Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6 --hdfsUser
impUser1 --spn adpqa@INFAKRB.INFADEV.COM --keytab /bdmqa/BDM_Automation/Source/
adpqa_AD.keytab --ccoConfPath /data/Informatica/10.2.2_252/tomcat/bin/disTemp/
inkrh71hdp07_252/DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-site.xml hdfs://nameservice1/
BDM_Automation/Target/HDFSTargetAppend/SparkMode/MultiPartition//
*_tmp_infa_7939778750618549156_StudentHDFSTargetAppend.out\n-- ----\n-- Command
[Pre_Spark_Task_Command_1_7] \n/data/Informatica/10.2.2_252/services/shared/hadoop/
HDP_2.6/scripts/HadoopFsMkdir --hadoop.home /data/Informatica/10.2.2_252/services/shared/
hadoop/HDP_2.6 --hdfsUser impUser1 --spn adpqa@INFAKRB.INFADEV.COM --keytab /bdmqa/
BDM_Automation/Source/adpqa_AD.keytab --ccoConfPath /data/Informatica/10.2.2_252/
tomcat/bin/disTemp/inkrh71hdp07_252/DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-site.xml hdfs://
nameservice1/BDM_Automation/Target/HDFSTargetAppend/SparkMode/MultiPartition//
_tmp_infa_7939778750618549156_StudentHDFSTargetAppend.out/\n",
            "depends": ""
        },
        {
            "name": "Post_Spark_Task_Command_1",
            "content": "-- ----\n-- Command [Post_Spark_Task_Command_1_1] \n/data/
Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6/scripts/FileCopyUtil --
hadoop.home /data/Informatica/10.2.2_252/services/shared/hadoop/HDP_2.6 --mergefiles --
deleteEmptyFiles --hdfsUser impUser1 --copyToLocal --spn adpqa@INFAKRB.INFADEV.COM --
keytab /bdmqa/BDM_Automation/Source/adpqa_AD.keytab --ccoConfPath /data/Informatica/
10.2.2_252/tomcat/bin/disTemp/inkrh71hdp07_252/DIS_HDP_2.6/cco_hdp_26/SPARK/infacco-
site.xml hdfs://nameservice1//tmp/SPARK_impUser1/sess7939778750618549156//
W7939778750618549156_Write_HDFSAppendTarget_HDFSTgtAppend_MultiPartition_SparkMode/
reject-files file:///data/Informatica/10.2.2_252/tomcat/bin/reject/.bad\n",
            "depends": "InfaSpark0"
        }
    ]
}
```

INDEX