Informatica®

Informatica® Cloud Data Integration

Databricks Connector

Informatica Cloud Data Integration Databricks Connector
July 2024

© Copyright Informatica LLC 2018, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, the Informatica logo, Informatica Cloud, and PowerCenter are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at https://www.informatica.com/trademarks.html. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

See patents at https://www.informatica.com/legal/patents.html.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

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: 2024-07-18

Table of Contents

Preface 6

Informatica Resources 6

- Informatica Documentation 6
- Informatica Intelligent Cloud Services web site 6
- Informatica Intelligent Cloud Services Communities 6
- Informatica Intelligent Cloud Services Marketplace 6
- Data Integration connector documentation 7
- Informatica Knowledge Base 7
- Informatica Intelligent Cloud Services Trust Center 7
- Informatica Global Customer Support 7

Chapter 1: Introduction to Databricks Connector 8

Databricks Connector assets 8

External table support in Databricks 8

Chapter 2: Connections for Databricks 10

Staging prerequisites 10

SQL warehouse 10

- Configure AWS staging 10
- Configure Azure staging 14

Databricks cluster 15

- Configure Spark parameters 15
- Configure Secure Agent properties 16

Connect to Databricks 16

- Before you begin 17
- Connection details 17
- Advanced settings 18

Configure proxy settings 23

JDBC URL parameters 23

Rules and guidelines for personal staging location 24

Private links to access Databricks 24

Chapter 3: Mappings for Databricks 25

Before you begin 25

- Verify permissions 26

Sources for Databricks 27

- Source properties for Databricks 28
- Custom query source type 31
- Key range partitioning 31

Targets for Databricks 32

Target properties for Databricks . . . 33
Create a target table at runtime. . . . 36
Override the update operation. . . . 37
Passthrough partitioning. . . . 38
Determine the order of processing for multiple targets. . . . 38
Optimize the staging performance of a mapping. . . . 39
Lookups for Databricks. . . . 40
Lookup properties for Databricks. . . . 40
Enable lookup caching. . . . 42
SQL transformation. . . . 42
Dynamic schema handling. . . . 43
Parameterization. . . . 44
IDENTITY columns . . . 44
Mappings in advanced mode example. . . . 45
Rules and guidelines for mappings. . . . 46
Rules and guidelines for mappings in advanced mode. . . . 47

Chapter 4: Migrating a mapping. . . . 50

Plan the migration. . . . 50
Migrate a mapping within the same path. . . . 50
Migrate a mapping to a different path. . . . 51
Migration options. . . . 51
Migration restrictions. . . . 52

Chapter 5: SQL ELT with Databricks Connector. . . . 53

SQL ELT configuration options. . . . 53
Create a mapping in SQL ELT mode. . . . 53
Enable SQL ELT optimization for a mapping task . . . 55
SQL ELT query preview. . . . 55
Mappings in SQL ELT mode for Databricks. . . . 55
Sources in mappings in SQL ELT mode. . . . 56
Targets in mappings in SQL ELT mode. . . . 56
Transformations in mappings in SQL ELT mode. . . . 57
Functions in mappings in SQL ELT mode. . . . 57
Operators in mappings in SQL ELT mode. . . . 62
Rules and guidelines in mappings in SQL ELT mode. . . . 62
SQL ELT optimization for mapping tasks. . . . 63
SQL ELT optimization types. . . . 63
Read from and write to Databricks. . . . 64
Read from Amazon S3 and write to Databricks. . . . 64
Read from Microsoft Azure Data Lake Storage Gen2 and write to Databricks. . . . 65
SQL ELT compatibility. . . . 65
Configuring SQL ELT optimization . . . 67

Transformations with Databricks. 68
Features. 70
Configuring a custom query for the Databricks source object. 73
SQL ELT optimization for multiple targets. 73
Single commit for SQL ELT optimization. 74
Rules and guidelines for SQL ELT optimization 74
Troubleshooting SQL ELT optimization. 78

Chapter 6: Data type reference. 79

Databricks and transformation data types. 79

Index. 81

Preface

Use *Databricks Connector* to learn how to read from or write to Databricks by using Cloud Data Integration. Learn to create a connection and develop mappings, mapping tasks, dynamic mapping tasks, and data transfer tasks in Cloud Data Integration.

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 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 Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

https://network.informatica.com/community/informatica-network/products/cloud-integration

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit https://docs.informatica.com.

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 Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, on the Informatica Intelligent Cloud Services Status page, click **SUBSCRIBE TO UPDATES**. You can choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

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

To find online support resources on the Informatica Network, click **Contact Support** in the Informatica Intelligent Cloud Services Help menu to go to the **Cloud Support** page. The **Cloud Support** page includes system status information and community discussions. Log in to Informatica Network and click **Need Help** to find additional resources and to contact Informatica Global Customer Support through email.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Introduction to Databricks Connector

You can use Databricks Connector to securely read data from or write data to Databricks.

When you use Databricks Connector, you can create a Databricks connection to connect to Databricks endpoints hosted on Amazon Web Services, and Microsoft Azure environment.

When you read data from and write data to Databricks, you can stage the data in Amazon Web Services, Microsoft Azure environment, or personal staging location.

You can use Databricks Connector on the Windows and Linux operating systems.

For Linux operating systems, you can switch mappings to advanced mode to include transformations and functions that enable advanced functionality. A mapping in advanced mode can run on the advanced cluster hosted on Amazon Web Services, Microsoft Azure environment, or on a self-service cluster.

Databricks Connector assets

Create assets in Data Integration to integrate data using Databricks Connector.

When you use Databricks Connector, you can include the following Data Integration assets:

- Data transfer task
- Dynamic mapping task
- Mapping
- Mapping task

For more information about configuring assets and transformations, see *Mappings*, *Transformations*, and *Tasks* in the Data Integration documentation.

External table support in Databricks

External tables store their data in locations outside of the predefined managed storage location associated with the metastore, unity catalog, or schema. An external table references an external storage path by using a `LOCATION` clause. For more information on external tables, see the Databricks documentation.

Note: You can read and write data to external tables of file type Delta in Databricks.

You can use the following data types when you create an external table:

- Array*
- Binary
- Bigint
- Boolean
- Date
- Decimal
- Double
- Float
- Int
- Map*
- Smallint
- String
- Struct*
- Tinyint
- Timestamp

*Applies only to mappings in advanced mode.

When you create an external table, specify the table location when you create a new target at runtime. For more information, see “Create a target table at runtime” on page 36

CHAPTER 2

Connections for Databricks

Create a Databricks connection to securely read data from or write data to Databricks.

You can use a Databricks connection to specify sources, targets, and lookups in mappings and mapping tasks.

Staging prerequisites

Before you create a connection, you must perform certain prerequisite tasks to configure the staging environment to connect to SQL warehouse or Databricks cluster.

SQL warehouse

Configure either the AWS or Azure staging environment for the SQL warehouse based on the deployed environment. You also need to configure the Spark parameters for the SQL warehouse to use Azure and AWS staging.

You can use a SQL warehouse on the Windows and Linux operating systems.

Configure AWS staging

Configure IAM AssumeRole authentication to use AWS staging for the SQL warehouse.

IAM AssumeRole authentication

You can enable IAM AssumeRole authentication in Databricks for secure and controlled access to the Amazon S3 staging bucket when you run mappings and mapping tasks.
You can configure IAM authentication when the Secure Agent runs on an Amazon Elastic Compute Cloud (EC2) system. When you use a serverless runtime environment, you cannot configure IAM authentication.

Perform the following steps to configure IAM authentication on EC2:

1. Create a minimal Amazon IAM policy.
2. Create the Amazon EC2 role. The Amazon EC2 role is used when you create an EC2 system.
   For more information about creating the Amazon EC2 role, see the *AWS documentation*.
3. Link the minimal Amazon IAM policy with the Amazon EC2 role.

4. Create an EC2 instance. Assign the Amazon EC2 role that you created to the EC2 instance.
5. Install the Secure Agent on the EC2 system.

Temporary security credentials using AssumeRole

You can use temporary security credentials using AssumeRole to access AWS resources from same or different AWS accounts.
Ensure that you have the **sts:AssumeRole** permission and a trust relationship established within the AWS accounts to use temporary security credentials. The trust relationship is defined in the trust policy of the IAM role when you create the role. The IAM role adds the IAM user as a trusted entity allowing the IAM users to use temporary security credentials and access AWS accounts.

For more information about how to establish the trust relationship, see the *AWS documentation*.

When the trusted IAM user requests for temporary security credentials, the AWS Security Token Service (AWS STS) dynamically generates the temporary security credentials that are valid for a specified period and provides the credentials to the trusted IAM users. The temporary security credentials consist of access key ID, secret access key, and secret token.

To use the dynamically generated temporary security credentials, provide a value for the **IAM Role ARN** connection property when you create a Databricks connection. The IAM Role ARN uniquely identifies the AWS resources. Then, specify the time duration in seconds during which you can use the temporarily security credentials in the **Temporary Credential Duration** advanced source and target properties.

External ID

You can specify the external ID for a more secure cross-account access to the Amazon S3 bucket when the Amazon S3 bucket is in a different AWS account.
Optionally, you can specify the external ID in the AssumeRole request to the AWS Security Token Service (STS).

The external ID must be a string.

The following sample shows an external ID condition in the assumed IAM role trust policy:

```
"Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AWS_Account_ID : user/user_name"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "dummy_external_id"
        }
      }
    }
  ]
```

Temporary security credentials policy

To use temporary security credentials to access AWS resources, both the IAM user and IAM role require policies.

Amazon S3 permission policy

Attach the following S3 permission policy to allow access to the Amazon S3 bucket:

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
            {
                "Action": [
                    "s3:DeleteObject",
                    "s3:GetObject",
                    "s3:ListBucket",
                    "s3:PutObject",
                    "s3:PutObjectTagging",
                    "s3:GetBucketAcl"
                ],
                "Effect": "Allow",
                "Resource": "arn:aws:s3:::com.amk"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:DeleteObject",
                    "s3:GetObject",
                    "s3:ListBucket",
                    "s3:PutObject",
                    "s3:PutObjectTagging",
                    "s3:GetBucketAcl"
                ],
                "Resource": "arn:aws:s3:::com.amk/*"
            }
        ]
    }
```

The following section lists the policies required for IAM user and IAM role:

IAM user

An IAM user must have the `sts:AssumeRole` policy to use temporary security credentials in same or different AWS account.

The following sample policy allows an IAM user to use the temporary security credentials in an AWS account:

```
{
"Version":"2012-10-17",
"Statement":{
"Effect":"Allow",
"Action":"sts:AssumeRole",
"Resource":"arn:aws:iam::<ACCOUNT-HYPHENS>:role/<ROLE-NAME>" }
}
```

IAM role

An IAM role must have the `sts:AssumeRole` policy and a trust policy attached with the IAM role to allow the IAM user to access the AWS resource using temporary security credentials. The policy specifies the AWS resource that the IAM user can access and the actions that the IAM user can perform. The trust policy specifies the IAM user from the AWS account that can access the AWS resource.

The following policy is a sample trust policy:

```
{
"Version":"2012-10-17",
"Statement":[
{
"Effect":"Allow",
"Principal":{ "AWS":"arn:aws:iam::AWS-account-ID:root" },
"Action":"sts:AssumeRole"
}
 ]
  }
   }
```

Here, in the `Principal` attribute, you can also provide the ARN of IAM user who can use the dynamically generated temporary security credentials and to restrict further access. For example,

```
"Principal" : { "AWS" : "arn:aws:iam:: AWS-account-ID :user/ user-name " }
```

Temporary security credentials using AssumeRole for EC2

You can use temporary security credentials using AssumeRole for an Amazon EC2 role to access AWS resources from same or different AWS accounts.
The Amazon EC2 role would be able to assume another IAM Role from the same or a different AWS account without requiring the permanent access key and secret key.

Consider the following prerequisites when you use temporary security credentials using AssumeRole for EC2:

- Install the Secure Agent on an AWS service such as Amazon EC2.
- The EC2 role attached to the AWS EC2 service does not need access to Amazon S3 but needs permission to assume another IAM role.
- The IAM role that needs to be assumed by the EC2 role must have a permission policy and a trust policy attached to it.

To configure an EC2 role to assume the IAM role provided in the **IAM Role ARN** connection property, select the **Use EC2 Role to Assume Role** check box in the connection properties.

Create a minimal Amazon IAM policy

To stage the data in Amazon S3, use the following minimum required permissions: :

- PutObject
- GetObject
- DeleteObject
- ListBucket
- ListBucketMultipartUploads. Applicable only for mappings in advanced mode.

You can use the following sample Amazon IAM policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:ListBucket",

"s3:ListBucketMultipartUploads"
                    ],
            "Resource": [
                "arn:aws:s3:::<bucket_name>/*",
                "arn:aws:s3:::<bucket_name>"
            ]
        }
    ]
}
```

For mappings in advanced mode, you can use different AWS accounts within the same AWS region. Make sure that the Amazon IAM policy confirms access to the AWS accounts used in these mappings.

Note: The **Test Connection** does not validate the IAM policy assigned to users. You can specify the Amazon S3 bucket name in the source and target advanced properties.

Configure Spark parameters for AWS staging

Before you use the Databricks SQL warehouse to run mappings, configure the Spark parameters for SQL warehouse on the Databricks SQL Admin console.

On the Databricks SQL Admin console, navigate to **SQL Warehouse Settings > Data Security**, and then configure the Spark parameters for AWS under **Data access configuration**.

Add the following Spark configuration parameters and restart the SQL warehouse:

- `spark.hadoop.fs.s3a.access.key <S3 Access Key value>`
- `spark.hadoop.fs.s3a.secret.key <S3 Secret Key value>`
- `spark.hadoop.fs.s3a.endpoint <S3 Staging Bucket endpoint value>`

For example, the S3 staging bucket warehouse value is *s3.ap-south-1.amazonaws.com*.

Ensure that the configured access key and secret key have access to the S3 buckets where you store the data for Databricks tables.

Configure Azure staging

Before you use Microsoft Azure Data Lake Storage Gen2 to stage files, perform the following tasks:

- Create a storage account to use with Microsoft Azure Data Lake Storage Gen2 and enable **Hierarchical namespace** in the Azure portal.
  You can use role-based access control to authorize the users to access the resources in the storage account. Assign the Contributor role or Reader role to the users. The contributor role grants you full access to manage all resources in the storage account, but does not allow you to assign roles. The reader role allows you to view all resources in the storage account, but does not allow you to make any changes.

  Note: To add or remove role assignments, you must have write and delete permissions, such as an Owner role.
- Register an application in Azure Active Directory to authenticate users to access the Microsoft Azure Data Lake Storage Gen2 account.
  You can use role-based access control to authorize the application. Assign the Storage Blob Data Contributor or Storage Blob Data Reader role to the application. The Storage Blob Data Contributor role lets you read, write, and delete Azure Storage containers and blobs in the storage account. The Storage Blob Data Reader role lets you only read and list Azure Storage containers and blobs in the storage account.
- Create an Azure Active Directory web application for service-to-service authentication with Microsoft Azure Data Lake Storage Gen2.

  Note: Ensure that you have superuser privileges to access the folders or files created in the application using the connector.
- To read and write complex files, set the JVM options for type DTM to increase the -Xms and -Xmx values in the system configuration details of the Secure Agent to avoid java heap space error. The recommended -Xms and -Xmx values are 512 MB and 1024 MB respectively.

Configure Spark parameters for Azure staging

Before you use the Databricks SQL warehouse to run mappings, configure the Spark parameters for SQL warehouse on the Databricks SQL Admin console.

On the Databricks SQL Admin console, navigate to **SQL Warehouse Settings > Data Security**, and then configure the Spark parameters for Azure under **Data access configuration**.

Add the following Spark configuration parameters and restart the SQL warehouse:

- `spark.hadoop.fs.azure.account.oauth2.client.id.<storage-account-name>.dfs.core.windows.net <value>`
- `spark.hadoop.fs.azure.account.auth.type.<storage-account-name>.dfs.core.windows.net OAuth`
- `spark.hadoop.fs.azure.account.oauth2.client.secret.<storage-account-name>.dfs.core.windows.net <Value>`
- `spark.hadoop.fs.azure.account.oauth.provider.type.<storage-account-name>.dfs.core.windows.net org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider`
- `spark.hadoop.fs.azure.account.oauth2.client.endpoint.<storage-account-name>.dfs.core.windows.net https://login.microsoftonline.com/<Tenant ID>/oauth2/token`

Ensure that the configured client ID and client secret have access to the file systems where you store the data for Databricks tables.

Databricks cluster

Configure the Spark parameters for the Databricks cluster to use Azure and AWS staging based on where the cluster is deployed.

You also need to enable the Secure Agent properties for runtime and design-time processing on the Databricks cluster.

You can use a Databricks cluster only on the Linux operating system.

Configure Spark parameters

Before you connect to the Databricks cluster, you must configure the Spark parameters on AWS and Azure.

Configuration on AWS

Add the following Spark configuration parameters for the Databricks cluster and restart the cluster:

- `spark.hadoop.fs.s3a.access.key <value>`
- `spark.hadoop.fs.s3a.secret.key <value>`
- `spark.hadoop.fs.s3a.endpoint <value>`

Ensure that the access and secret key configured has access to the buckets where you store the data for Databricks tables.

Configuration on Azure

Add the following Spark configuration parameters for the Databricks cluster and restart the cluster:

- `fs.azure.account.oauth2.client.id.<storage-account-name>.dfs.core.windows.net <value>`
- `fs.azure.account.auth.type.<storage-account-name>.dfs.core.windows.net <value>`
- `fs.azure.account.oauth2.client.secret.<storage-account-name>.dfs.core.windows.net <Value>`
- `fs.azure.account.oauth.provider.type.<storage-account-name>.dfs.core.windows.net org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider`
- `fs.azure.account.oauth2.client.endpoint.<storage-account-name>.dfs.core.windows.net https://login.microsoftonline.com/<Tenant ID>/oauth2/token`

Ensure that the client ID and client secret configured has access to the file systems where you store the data for Databricks tables.

Configure Secure Agent properties

When you configure mappings, the SQL warehouse processes the mapping by default.
To process the mappings on Databricks cluster, enable the Secure Agent properties.

To connect to all-purpose cluster and job cluster, enable the Secure Agent properties for design time and runtime respectively.

Setting the property for design time processing

Before you can import metadata and design mappings or mappings in advanced mode, perform the following steps:

1. In **Administrator**, select the Secure Agent listed on the **Runtime Environments** tab.
2. Click **Edit**.
3. In the **System Configuration Details** section, select Data Integration Server as the **Service** and Tomcat JRE as the **Type**.
4. Edit the **JRE_OPTS** field and set the value to *-DUseDatabricksSql=false*.



Setting the property for runtime processing

1. In **Administrator**, select the Secure Agent listed on the **Runtime Environments** tab.
2. Click **Edit**.
3. In the **System Configuration Details** section, select Data Integration Server as the **Service** and DTM as the **Type**.
4. Edit the **JVMOption** field.
   a. To run mappings, set the value to *-DUseDatabricksSql=false*.



   b. To run mappings enabled with SQL ELT optimization, set the value to *-DUseDatabricksSqlForPdo=false*.



Connect to Databricks

Let's configure the Databricks connection properties to connect to Databricks.

Before you begin

You can use a Databricks connection to connect to SQL warehouse or Databricks cluster to read from and write to Databricks tables.

- **SQL warehouse**
  The Secure Agent connects to the SQL warehouse at design time and runtime.
- **Databricks cluster**
  The term Databricks cluster refers to the all-purpose cluster and job cluster. The Secure Agent connects to the all-purpose cluster to import the metadata at design time and to the job cluster to run the mappings.

Before you get started, you'll need to get information from your Databricks account.

The following video shows you how to get the information you need:



You also need to configure the AWS or Azure staging environment to use the SQL warehouse or the Databricks cluster in the connection.

To learn about the staging prerequisites for the Azure or AWS environment, check out "SQL warehouse " on page 10 or "Databricks cluster" on page 15.

Connection details

The following table describes the basic connection properties:

Property	Description
Connection Name	Name of the connection. Each connection name must be unique within the organization. Connection names can contain alphanumeric characters, spaces, and the following special characters: _ . + -, Maximum length is 255 characters.
Description	Description of the connection. Maximum length is 4000 characters.
Type	Databricks
Runtime Environment	The name of the runtime environment where you want to run tasks. Select a Secure agent, Hosted Agent, or serverless runtime environment. Hosted Agent is not applicable for mappings in advanced mode.

Property	Description

SQL Warehouse JDBC URL	Databricks SQL Warehouse JDBC connection URL.
	Required to connect to a Databricks SQL warehouse. Doesn't apply to Databricks clusters.
	Note: Databricks SQL Serverless is the recommended Databricks cluster type.
	To get the SQL Warehouse JDBC URL, go to the Databricks console and select the JDBC driver version from the JDBC URL menu.
	For JDBC URL version 2.6.22 or earlier, use the following syntax:
	`jdbc:spark://<Databricks Host>:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/endpoints/<SQL endpoint cluster ID>;`
	For JDBC URL version 2.6.25 or later, use the following syntax:
	`jdbc:databricks://<Databricks Host>:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/endpoints/<SQL endpoint cluster ID>;`
	Ensure that you set the required environment variables in the Secure Agent. Also specify the correct **JDBC Driver Class Name** under advanced connection settings.
	Note: Specify the database name in the Database Name connection property. If you specify the database name in the JDBC URL, it is not considered. The Databricks Host, Organization ID, and Cluster ID properties are not considered if you configure the SQL warehouse JDBC URL property.

Databricks Token	Personal access token to access Databricks. Required for SQL warehouse and Databricks cluster. Ensure that you have permissions to attach to the cluster identified in the **Cluster ID** property. For mappings, you must have additional permissions to create Databricks clusters.
Catalog Name	If you use Unity Catalog, the name of an existing catalog in the metastore. Optional for SQL warehouse. Doesn't apply to Databricks cluster. You can also specify the catalog name in the end of the SQL warehouse JDBC URL. **Note:** The catalog name cannot contain special characters. For more information about Unity Catalog, see the Databricks documentation.

Advanced settings

The following table describes the advanced connection properties:

Property	Description
Database	The name of the schema in the unity catalog in Databricks. Optional for SQL warehouse and Databricks cluster. For Data Integration, if you do not specify a value, all databases available in the workspace are listed. The value you specify overrides the schema specified in the **SQL Warehouse JDBC URL** connection property.
JDBC Driver Class Name	The name of the JDBC driver class. Optional for SQL warehouse and Databricks cluster. For JDBC URL versions 2.6.22 or earlier, specify the driver class name as `com.simba.spark.jdbc.Driver`. For JDBC URL versions 2.6.25 or later, specify the driver class name as `com.databricks.client.jdbc.Driver`.

Property	Description
Staging Environment	The cloud provider where the Databricks cluster is deployed. Required for SQL warehouse and Databricks cluster. Select one of the following options: - AWS - Azure - Personal Staging Location Default is Personal Staging Location. You can select the Personal Staging Location as the staging environment instead of Azure or AWS staging environments to stage data locally for mappings and tasks. Personal staging location doesn't apply to Databricks cluster. You cannot use personal staging location when you configure mappings in advanced mode. **Note:** You cannot switch between clusters once you establish a connection.
Databricks Host	The host name of the endpoint the Databricks account belongs to. Required for Databricks cluster. Doesn't apply to SQL warehouse. You can get the Databicks Host from the JDBC URL. The URL is available in the Advanced Options of JDBC or ODBC in the Databricks all-purpose cluster. The following example shows the Databicks Host in JDBC URL: jdbc:spark://**<Databricks Host>**:443/ default;transportMode=http; ssl=1;httpPath=sql/ protocolv1/o/<Org Id>/<Cluster ID>; AuthMech=3; UID=token; PWD=<personal-access-token> The value of PWD in Databricks Host, Organization Id, and Cluster ID is always `<personal-access-token>`.
Cluster ID	The ID of the cluster. Required for Databricks cluster. Doesn't apply to SQL warehouse. You can get the cluster ID from the JDBC URL. The URL is available in the Advanced Options of JDBC or ODBC in the Databricks all-purpose cluster The following example shows the Cluster ID in JDBC URL: jdbc:spark://<Databricks Host>:443/ default;transportMode=http; ssl=1;httpPath=sql/ protocolv1/o/<Org Id>/**<Cluster ID>**; AuthMech=3;UID=token; PWD=<personal-access-token>
Organization ID	The unique organization ID for the workspace in Databricks. Required for Databricks cluster. Doesn't apply to SQL warehouse. You can get the Organization ID from the JDBC URL. The URL is available in the Advanced Options of JDBC or ODBC in the Databricks all-purpose cluster The following example shows the Organization ID in JDBC URL: jdbc:spark://<Databricks Host>:443/ default;transportMode=http; ssl=1;httpPath=sql/ protocolv1/o/**<Organization ID>**/ <Cluster ID>;AuthMech=3;UID=token; PWD=<personal-access-token>
Min Workers[1]	The minimum number of worker nodes to be used for the Spark job. Minimum value is 1. Required for Databricks cluster. Doesn't apply to SQL warehouse.
Max Workers[1]	The maximum number of worker nodes to be used for the Spark job. If you don't want to autoscale, set Max Workers = Min Workers or don't set Max Workers. Optional for Databricks cluster. Doesn't apply to SQL warehouse.
DB Runtime Version[1]	The version of Databricks cluster to spawn when you connect to Databricks cluster to process mappings. Required for Databricks cluster. Doesn't apply to SQL warehouse. Select the Databricks runtime version 9.1 LTS or 13.3 LTS.

Property	Description
Worker Node Type[1]	The worker node instance type that is used to run the Spark job. Required for Databricks cluster. Doesn't apply to SQL warehouse. For example, the worker node type for AWS can be i3.2xlarge. The worker node type for Azure can be Standard_DS3_v2.
Driver Node Type[1]	The driver node instance type that is used to collect data from the Spark workers. Optional for Databricks cluster. Doesn't apply to SQL warehouse. For example, the driver node type for AWS can be i3.2xlarge. The driver node type for Azure can be Standard_DS3_v2. If you don't specify the driver node type, Databricks uses the value you specify in the worker node type field.
Instance Pool ID[1]	The instance pool ID used for the Spark cluster. Optional for Databricks cluster. Doesn't apply to SQL warehouse. If you specify the Instance Pool ID to run mappings, the following connection properties are ignored: - Driver Node Type - EBS Volume Count - EBS Volume Type - EBS Volume Size - Enable Elastic Disk - Worker Node Type - Zone ID
Elastic Disk[1]	Enables the cluster to get additional disk space. Optional for Databricks cluster. Doesn't apply to SQL warehouse. Enable this option if the Spark workers are running low on disk space.
Spark Configuration[1]	The Spark configuration to use in the Databricks cluster. Optional for Databricks cluster. Doesn't apply to SQL warehouse. The configuration must be in the following format: `"key1"="value1";"key2"="value2";...` For example, `"spark.executor.userClassPathFirst"="False"`
Spark Environment Variables[1]	The environment variables to export before launching the Spark driver and workers. Optional for Databricks cluster. Doesn't apply to SQL warehouse. The variables must be in the following format: `"key1"="value1";"key2"="value2";...` For example, `"MY_ENVIRONMENT_VARIABLE"="true"`
[1]Doesn't apply to mappings in advanced mode.	

AWS staging environment

The following table describes the properties for the AWS staging environment:

Property	Description
S3 Authentication Mode	The authentication mode to access Amazon S3. Select one of the following authentication modes: - Permanent IAM credentials. Uses the S3 access key and S3 secret key to connect to Databricks. - IAM Assume Role[1]. Uses the AssumeRole for IAM authentication to connect to Databricks. Doesn't apply to Databricks cluster.
S3 Access Key	The key to access the Amazon S3 bucket.
S3 Secret Key	The secret key to access the Amazon S3 bucket.
S3 Data Bucket	The existing bucket to store the Databricks data.
S3 Staging Bucket	The existing bucket to store the staging files.
S3 VPC Endpoint Type	The type of Amazon Virtual Private Cloud endpoint for Amazon S3. You can use a VPC endpoint to enable private communication with Amazon S3. Select one of the following options: - None. Select if you do not want to use a VPC endpoint. - Interface Endpoint. Select to establish private communication with Amazon S3 through an interface endpoint which uses a private IP address from the IP address range of your subnet. It serves as an entry point for traffic destined to an AWS service.
Endpoint DNS Name for S3	The DNS name for the Amazon S3 interface endpoint. Replace the asterisk symbol with the bucket keyword in the DNS name. Enter the DNS name in the following format: `bucket.<DNS name of the interface endpoint>` For example, `bucket.vpce-s3.us-west-2.vpce.amazonaws.com`
IAM Role ARN[1]	The Amazon Resource Number (ARN) of the IAM role assumed by the user to use the dynamically generated temporary security credentials. Set the value of this property if you want to use the temporary security credentials to access the Amazon S3 staging bucket. For more information about how to get the ARN of the IAM role, see the *AWS documentation*.
Use EC2 Role to Assume Role[1]	Optional. Select the check box to enable the EC2 role to assume another IAM role specified in the IAM Role ARN option. The EC2 role must have a policy attached with a permission to assume an IAM role from the same or different AWS account.
STS VPC Endpoint Type	The type of Amazon Virtual Private Cloud endpoint for AWS Security Token Service. You can use a VPC endpoint to enable private communication with Amazon Security Token Service. Select one of the following options: - None. Select if you do not want to use a VPC endpoint. - Interface Endpoint. Select to establish private communication with Amazon Security Token Service through an interface endpoint which uses a private IP address from the IP address range of your subnet.

Property	Description
Endpoint DNS Name for AWS STS	The DNS name for the AWS STS interface endpoint. For example, `vpce-01f22cc14558c241f-s8039x4c.sts.us-west-2.vpce.amazonaws.com`
S3 Service Regional Endpoint	The S3 regional endpoint when the S3 data bucket and the S3 staging bucket need to be accessed through a region-specific S3 regional endpoint. Doesn't apply to Databricks cluster. Default is `s3.amazonaws.com`.
S3 Region Name[1]	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.
Zone ID[1]	The zone ID for the Databricks job cluster. Optional for Databricks cluster. Doesn't apply to SQL warehouse. Applies only if you want to create a Databricks job cluster in a particular zone at runtime. For example, us-west-2a. **Note:** The zone must be in the same region where your Databricks account resides.
EBS Volume Type[1]	The type of EBS volumes launched with the cluster. Optional for Databricks cluster. Doesn't apply to SQL warehouse.
EBS Volume Count[1]	The number of EBS volumes launched for each instance. You can choose up to 10 volumes. Optional for Databricks cluster. Doesn't apply to SQL warehouse. **Note:** In a Databricks connection, specify at least one EBS volume for node types with no instance store. Otherwise, cluster creation fails.
EBS Volume Size[1]	The size of a single EBS volume in GiB launched for an instance. Optional for Databricks cluster. Doesn't apply to SQL warehouse.
[1]Doesn't apply to mappings in advanced mode.	

Azure staging environment

The following table describes the properties for the Azure staging environment:

Property	Description
ADLS Storage Account Name	The name of the Microsoft Azure Data Lake Storage account.
ADLS Client ID	The ID of your application to complete the OAuth Authentication in the Active Directory.
ADLS Client Secret	The client secret key to complete the OAuth Authentication in the Active Directory.
ADLS Tenant ID	The ID of the Microsoft Azure Data Lake Storage directory that you use to write data.
ADLS Endpoint	The OAuth 2.0 token endpoint from where authentication based on the client ID and client secret is completed.

Property	Description
ADLS Filesystem Name	The name of an existing file system to store the Databricks data.
ADLS Staging Filesystem Name	The name of an existing file system to store the staging data.

Configure proxy settings

If your organization uses an outgoing proxy server to connect to the Internet, the agent connects to Informatica Intelligent Cloud Services through the proxy server.

You can configure the Secure Agent to use the proxy server on Windows and Linux.

You can use only an unauthenticated proxy server to connect to Informatica Intelligent Cloud Services. You can configure proxy settings only when you use the AWS staging environment.

You cannot configure proxy settings for mappings in advanced mode.

To configure the proxy settings for the Secure Agent, perform the following tasks:

- Configure the Secure Agent through the Secure Agent Manager on Windows or shell command on Linux. For instructions, see "Configure the proxy settings on Windows" or "Configure the proxy settings on Linux," in *Getting Started* in the Data Integration documentation.
- Configure the JVM options for the DTM in the Secure Agent properties. For instructions, see the Proxy server settings Knowledge Base article.

Note: If you enable both HTTP and SOCKS proxies, SOCKS proxy is used by default. If you want to use HTTP proxy instead of SOCKS proxy, set the value of the DisableSocksProxy property to true in the System property.

To configure the proxy settings for the serverless runtime environment, see *Runtime Environments* in the Administrator help.

JDBC URL parameters

You can utilize the additional JDBC URL parameters field in the Databricks connection to customize and set any additional parameters required to connect to Databricks.
You can configure the following properties as additional JDBC URL parameters in the Databricks connection:

- To pass the unity catalog information to Databricks, specify the catalog name after the SQL warehouse cluster ID in the following format:
  `jdbc:spark://<Databricks Host>:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/endpoints/<SQL endpoint cluster ID>;ConnCatalog=<catalog_name>;`
- To connect to Databricks using the proxy server, enter the following parameters:
  `jdbc: spark://<Databricks Host>:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/warehouses/`

219fe3013963cdce;**UseProxy=<Proxy=true>;ProxyHost=<proxy host IPaddress>;ProxyPort=<proxy server port number>;ProxyAuth=<Auth_true>;**

- To connect to SSL-enabled Databricks, specify the value in the JDBC URL in the following format: `jdbc:spark://<Databricks Host>:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/endpoints/<SQL endpoint cluster ID>;`

Rules and guidelines for personal staging location

When you select the personal staging location as a staging environment, the data is first staged in a java temporary location and then copied to a personal staging location of the unity catalog. Both the staged files will be deleted after the mapping runs successfully.
However, to stage the data in a different directory, configure the DTM property `-Djava.io.tmpdir=/my/dir/path` in the JVM options in the system configuration settings of the Administrator service.

To enable data staging in a different directory, you should have read and write permission and enough disk space to stage the data in the directory.

When you specify a personal staging location in the Databricks connection properties for staging, consider the following rules and guidelines:

- You can only specify unity enabled catalog in the SQL warehouse JDBC URL.
- All mappings that are configured run without SQL ELT optimization.
- The data is staged in the folder `stage://tmp/<user_name>` where the `<user_name>` is picked from the Databricks token provided in the connection and requires read and write access to the personal staging location in root location of AWS and Azure.

Private links to access Databricks

You can access Databricks using Azure Private Link endpoints.

To connect to the Databricks account over the private Azure network, see Secure connectivity to Azure Data Services.

CHAPTER 3

Mappings for Databricks

When you configure a mapping, you describe the flow of data from the source to the target.

A mapping defines reusable data flow logic that you can use in mapping tasks.

When you create a mapping, you define the Source, Target, and Lookup transformations to represent a Databricks object. Use the Mapping Designer in Data Integration to add the Source, Target, or Lookup transformations in the mapping canvas and configure the Databricks source, target, and lookup properties.

After you create a mapping, you can run the mapping or you can deploy the mapping in a mapping task. The mapping task allows you to process data based on the data flow logic defined in a mapping.

In advanced mode, the Mapping Designer updates the mapping canvas to include transformations and functions that enable advanced functionality.

You can use Monitor to monitor the jobs.

The following table lists the transformations that are supported by SQL warehouse and Databricks cluster:

Property	SQL warehouse[1]	Databricks cluster[2]
Source	Yes	Yes
Target	Yes	Yes
Filter	Yes	Yes
Lookup	Yes	No
Sorter	Yes	Yes
SQL	Yes	No

[1]The Secure Agent connects to the SQL warehouse at design time and runtime.

[2]The Secure Agent connects to the all-purpose cluster to import the metadata at design time and to the job cluster to run the mappings.

Before you begin

Before you configure and run a mapping, complete the required prerequisites.

Verify permissions

Permissions define the level of access for the operations that you can perform in Databricks.

The following table lists the permission that you need to run SQL commands to perform specific operations in Databricks:

SQL command	Description
CREATE TABLE	Create a managed table.
CREATE VIEW	Create a view.
CREATE TABLE ... LOCATION <EXTERNAL LOCATION>	Create an external table.
DROP TABLE	Delete a table.
DROP VIEW	Delete a view.
ALTER TABLE	Add a column.
SELECT FROM	Read from a table.
INSERT INTO	Use Insert operation with Write Disposition option set to Append.
INSERT OVERWRITE INTO	Use Insert operation with Write Disposition option set to Truncate.
MERGE INTO	Use Update, Upsert, and Delete operations.
DELETE FROM	Empty a table.
COPY INTO	Load data into target (ecosystem SQL ELT with CSV as source).
SELECT ON FILE	Retrieve data from a file staged in either Amazon S3, ADLS Gen2, or a personal staging location using the SELECT FROM command. Here are examples of the staging locations where you use the SELECT FROM command to retrieve data: Read from a file in Amazon S3: `INSERT INTO `tgt_table` (col1, col2) SELECT col1, col2 FROM parquet.`s3a://staging.bucket/infa_tmp-20231120_062134_910212391`` Read from a file in ADLS Gen2: `INSERT INTO `tgt_table` (col1, col2) SELECT col1, col2 FROM parquet.`abfss://blob@storage_account.dfs.core.windows.net/infa_tmp-20231120_070825_17050631/`` Read from a file in personal staging location: `INSERT INTO `tgt_table` (col1, col2) SELECT col1, col2 FROM parquet.`stage://tmp/username/infa_tmp-20231120_025852_313310201``

Personal staging location

To use the personal staging location as a staging environment, ensure that you have access to the personal staging location APIs.

Sources for Databricks

Add a Source transformation to extract data from a source.

When you add a Source transformation to a mapping, you define the source connection, source objects, and source properties related to the Databricks connection type.

The following table lists the source properties that are supported by SQL warehouse and Databricks cluster:

Property	SQL warehouse[1]	Databricks cluster[2]
Source Type - Single Object	Yes	Yes
Source Type - Parameter	Yes	Yes
Source Type - Query	Yes	No
Source Type - Multiple Objects	Yes	No
Parameter	Yes	Yes
Filter	Yes	Yes
Sort	No	No
Select distinct rows only	No	No
Database Name	Yes	Yes
Table Name	Yes	Yes
Pre SQL	Yes	No
Post SQL	Yes	No
SQL Override	Yes	No
Staging Location	Yes	Yes
Job Timeout	No	Yes. Applies only to job cluster.
Job Status Poll Interval	No	Yes. Applies only to job cluster.
DB REST API Timeout	No	Yes. Applies only to job cluster.
DB REST API Retry Interval	No	Yes. Applies only to job cluster.
Tracing Level	Yes	Yes
Key Range Partitioning	Yes	No

[1]The Secure Agent connects to the SQL warehouse at design time and runtime.

[2]The Secure Agent connects to the all-purpose cluster to import the metadata at design time and to the job cluster to run the mappings.

Source properties for Databricks

In a mapping, you can configure a Source transformation to represent a Databricks object.

The following table describes the Databricks source properties that you can configure in a Source transformation:

Property	Description
Connection	Name of the source connection. Select a source connection or click **New Parameter** to define a new parameter for the source connection. **Note:** You can completely paramaterize a parameter file for a source connection only for a single object source type. Parameterization doesn't apply to mappings in advanced mode.
Source Type	Type of the source object. Select any of the following source objects: - Single Object - Multiple Objects[1]. You can only use advanced relationships with multiple objects. - Query. - Parameter[1]. Select **Parameter** to define the source type when you configure the task. Multiple objects and query source types don't apply to Databricks cluster. **Note:** Multi-object database override will override the database for all imported objects, while the table override will only override the first table of the multi-object source.
Object	Name of the source object. You cannot use the data preview option if the source fields contain hierarchical data types.
Query[1]	Click on **Define Query** and enter a valid custom query. The **Query** property appears only if you select **Query** as the source type. You can parameterize a custom query object at runtime in a mapping. You can also enable unity catalog settings in a custom query to access a table within a particular catalog.
[1]Doesn't apply to mappings in advanced mode.	

The following table describes the Databricks query options that you can configure in a Source transformation:

Property	Description
Query Options	Filters the source data based on the conditions you specify. Click **Configure** to configure a filter option. The Filter option filters records and reduces the number of rows that the Secure Agent reads from the source. Add conditions in a read operation to filter records from the source. You can specify the following filter conditions: - Not parameterized. Use a basic filter to specify the object, field, operator, and value to select specific records. - Completely parameterized*. Use a parameter to specify the filter query. - Advanced. Use an advanced filter to define a complex filter condition. **Note:** You can use Contains, Ends With, and Starts With operators to filter records only on SQL endpoints.
Filter	Filters records based on the filter condition. You can specify a simple filter or an advanced filter.

The following table describes the Databricks advanced source properties that you can configure in a Source transformation:

Property	Description
Database Name	Overrides the database name provided in connection and the database name provided during metadata import. **Note:** To read from multiple objects ensure that you have specified the database name in the connection properties.
Table Name	Overrides the table name used in the metadata import with the table name that you specify.
Pre SQL	The pre-SQL command to run on the Databricks source table before the agent reads the data. Doesn't apply to Databricks cluster. For example, if you want to update records in the database before you read the records from the table, specify a pre-SQL statement. The query must include a fully qualified table name. You can specify multiple pre-SQL commands, each separated with a semicolon.
Post SQL	The post-SQL command to run on the Databricks table after the agent completes the read operation. Doesn't apply to Databricks cluster. For example, if you want to delete some records after the latest records are loaded, specify a post-SQL statement. The query must include a fully qualified table name. You can specify multiple post-SQL commands, each separated with a semicolon.

Property	Description
SQL Override	Overrides the default SQL query used to read data from Databricks custom query source. The column names in the SQL override query should match with the column names in the custom query in a SQL transformation. **Note:** The metadata of the source should be the same as SQL override to override the query.
Staging Location	Relative directory path to store the staging files. - If the Databricks cluster is deployed on AWS, use the path relative to the Amazon S3 staging bucket. - If the Databricks cluster is deployed on Azure, use the path relative to the Azure Data Lake Store Gen2 staging filesystem name. **Note:** When you use the unity catalog, a pre-existing location on user's cloud storage must be provided in the Staging Location.
Job Timeout	Maximum time in seconds that is taken by the Spark job to complete processing. Doesn't apply to SQL warehouse. If the job is not completed within the time specified, the Databricks cluster terminates the job and the mapping fails. If the job timeout is not specified, the mapping shows success or failure based on the job completion.
Job Status Poll Interval	Poll interval in seconds at which the Secure Agent checks the status of the job completion. Doesn't apply to SQL warehouse. Default is 30 seconds.
DB REST API Timeout	The Maximum time in seconds for which the Secure Agent retries the REST API calls to Databricks when there is an error due to network connection or if the REST endpoint returns `5xx HTTP` error code. Doesn't apply to SQL warehouse. Default is 10 minutes.
DB REST API Retry Interval	The time Interval in seconds at which the Secure Agent must retry the REST API call, when there is an error due to network connection or when the REST endpoint returns `5xx HTTP` error code. Doesn't apply to SQL warehouse. This value does not apply to the Job status REST API. Use job status poll interval value for the Job status REST API. Default is 30 seconds.
Tracing Level	Sets the amount of detail that appears in the log file. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.

Note: Advanced source properties are not applicable to mappings in advanced mode. Only pre-SQL and post-SQL advanced properties are applicable for custom queries.

Custom query source type

You can use a custom query as a source object when you use a Databricks connection.

You might want to use a custom query as the source when a source object is large. You can use the custom query to reduce the number of fields that enter the data flow. You can also create a parameter for the source type when you design your mapping so that you can define the query in the Mapping Task wizard.

To use a custom query as a source, select **Query** as the source type when you configure the source transformation and then use valid and supported SQL to define the query.

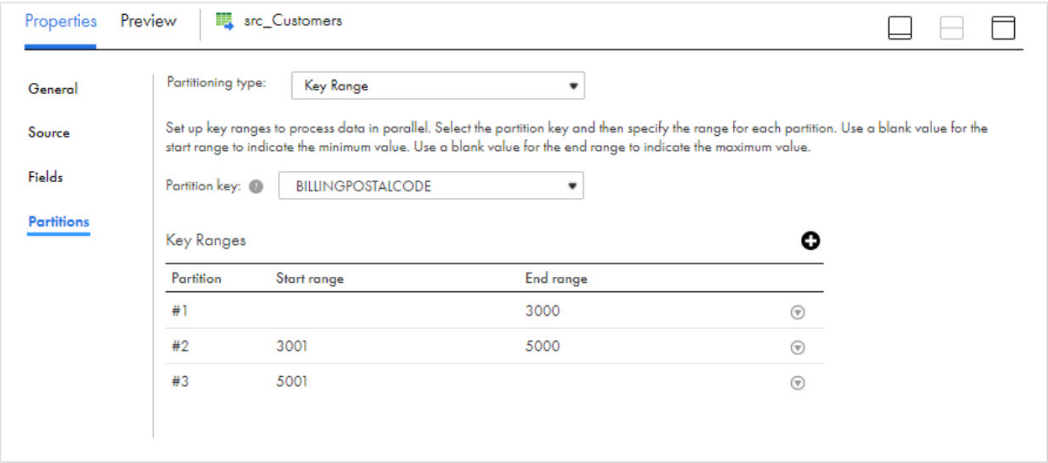
Key range partitioning

You can configure key range partitioning when you use a mapping to read data from a single Databricks source. The partition type controls how the agent distributes data among partitions at partition points. You cannot use partitioning in mappings in advanced mode.

Partitioning optimizes the mapping performance at run time. When you run a mapping configured with key range partitioning, the agent distributes rows of source data based on the field that you define as partition keys. The agent compares the field value to the range values for each partition and sends rows to the appropriate partitions.

On the **Partitions** tab for the Source transformation, select key range partitioning and choose the field as the partition key.

You can add key ranges to create partitions, as shown in the following image:



You can configure a partition key for fields of the following data types:

- Int, Bigint, Smallint, and Tiny int
- Timestamp
- Date. Specify the date and time in the following format: YYYY-MM-DD HH24:MI:SS.
- Decimal and Double. Specify the range values as whole numbers.

Rules and guidelines for key range partitioning

Consider the following rules and guidelines when you use key range partitioning:

- When you use key range partitioning, you can't perform the following operations:
  - Run a mapping in Databricks cluster.
  - Use Query or Multiple Objects as the source type.

- Configure SQL ELT optimization.
- Configure a SQL override.

- When a column configured as partition key contains null values, the null values are not written to the target.

Targets for Databricks

Add a Target transformation to write data to a target.

When you add a Target transformation to a mapping, you define the target connection, target objects, and target properties related to the Databricks connection type.

The following table lists the target properties that are supported by SQL warehouse and Databricks cluster:

Property	SQL warehouse[1]	Databricks cluster[2]
Source Type - Single Object	Yes	Yes
Source Type - Parameter	Yes	Yes
Object - Existing	Yes	Yes
Object - Create New at Runtime	Yes	Yes
Create target - Object Name, Table Location, Database Name	Yes	Yes
Create target - Table Properties	Yes	No
Operation - Insert, Update, Upsert, Delete	Yes	Yes
Operation - Data Driven	Yes	No
Target Database Name	Yes	Yes
Target Table Name	Yes	Yes
Update Override Query	Yes	No
Write Disposition - Append, Truncate	Yes	Yes
Write Disposition - Truncate Always	Yes	No
Update Mode - Update as update, Update else insert	Yes	Yes
Staging Location	Yes	Yes
Pre SQL	Yes	No
Post SQL	Yes	No
DTM Staging File Size	Yes	No

Property	SQL warehouse[1]	Databricks cluster[2]
Job Timeout	No	Yes. Applies only to job cluster.
Job Status Poll Interval	No	Yes. Applies only to job cluster.
DB REST API Timeout	No	Yes. Applies only to job cluster.
DB REST API Retry Interval	No	Yes. Applies only to job cluster.
Forward Rejected Rows	Yes	Yes

[1]The Secure Agent connects to the SQL warehouse at design time and runtime.

[2]The Secure Agent connects to the all-purpose cluster to import the metadata at design time and to the job cluster to run the mappings.

Target properties for Databricks

In a mapping, you can configure a Target transformation to represent a Databricks object.

The following table describes the Databricks properties that you can configure in a Target transformation:

Property	Description
Connection	Name of the target connection. Select a target connection or click **New Parameter** to define a new parameter for the target connection.
Target Type	Target type. Select one of the following types: - Single Object. - Parameter. Select **Parameter** to define the target type when you configure the task.
Object	Name of the target object.
Create Target	Creates a target. Enter a name for the target object and select the source fields that you want to use. By default, all source fields are used. You can select an existing target object or create a new target object at runtime. You cannot parameterize the target at runtime.
Operation	Defines the type of operation to be performed on the target table. Select from the following list of operations: - Insert (Default) - Update - Upsert - Delete - Data Driven[1] When you use an upsert operation, you must configure the **Update Mode** in target details as Update else Insert. If the key column gets null value from the source, the following actions take place for different operations: - Update. Skips the operation and does not update the row. - Delete. Skips the operation and does not delete the row. - Upsert. Inserts a new row instead of updating the existing row.

Property	Description
Update Columns	The fields to use as temporary primary key columns when you update, upsert, or delete data on the Databricks Delta target tables. When you select more than one update column, the mapping task uses the AND operator with the update columns to identify matching rows. Applies to update, upsert, delete and data driven operations.
Data Driven Condition[1]	Flags rows for an insert, update, delete, or reject operation based on the expressions that you define. For example, the following IIF statement flags a row for reject if the ID field is null. Otherwise, it flags the row for update: `IIF (ISNULL(ID), DD_REJECT, DD_UPDATE )` Required if you select the data driven operation.
[1]Applies only to mappings in advanced mode.	

The following table describes the Databricks advanced properties that you can configure in a Target transformation:

Advanced Property	Description
Target Database Name[1]	Overrides the database name provided in the connection and the database selected in the metadata browser for existing targets. **Note:** You cannot override the database name when you create a new target at runtime.
Target Table Name[1]	Overrides the table name at runtime for existing targets.
Update Override Query	Overrides the default update query that the agent generates for the update operation specified in this field. Doesn't apply to Databricks cluster. Use the merge command for the update operation.
Write Disposition	Overwrites or adds data to the existing data in a table. You can select from the following options: - Append. Appends data to the existing data in the table even if the table is empty. - Truncate. Overwrites the existing data in the table. Only applies to Insert operation and non-empty sources. - Truncate Always. Overwrites the existing data in the table. Applies to insert, update, upsert, and delete target operations for empty and non-empty sources. Doesn't apply to Databricks cluster. **Note:** You cannot perform `Truncate Always` when you don't specify the database name in the connection properties and while creating a new target at runtime.
Update Mode[1]	Defines how rows are updated in the target tables. Select from the following options: - Update As Update: Rows matching the selected update columns are updated in the target. - Update Else Insert: Rows matching the selected update columns are updated in the target. Rows that don't match are appended to the target.
Staging Location	Relative directory path to store the staging files. - If the Databricks cluster is deployed on AWS, use the path relative to the Amazon S3 staging bucket. - If the Databricks cluster is deployed on Azure, use the path relative to the Azure Data Lake Store Gen2 staging filesystem name. **Note:** When you use the unity catalog, a pre-existing location on user's cloud storage must be provided in the Staging Location.

Advanced Property	Description
Pre SQL	The pre-SQL command to run before the agent writes to Databricks. For example, if you want to assign sequence object to a primary key field of the target table before you write data to the table, specify a pre-SQL statement. Doesn't apply to Databricks cluster. You can specify multiple pre-SQL commands, each separated with a semicolon.
Post SQL	The post-SQL command to run after the agent completes the write operation. For example, if you want to alter the table created by using create target option and assign constraints to the table before you write data to the table, specify a post-SQL statement. Doesn't apply to Databricks cluster. You can specify multiple post-SQL commands, each separated with a semicolon.
DTM Staging File Size	The size of the flat file that Data Integration creates locally in a temporary folder to stage the data before writing to Databrick. Doesn't apply to Databricks cluster. Default is 50 MB.
Job Timeout[1]	Maximum time in seconds that is taken by the Spark job to complete processing. Doesn't apply to SQL warehouse. If the job is not completed within the time specified, the Databricks cluster terminates the job and the mapping fails. If the job timeout is not specified, the mapping shows success or failure based on the job completion.
Job Status Poll Interval[1]	Poll interval in seconds at which the Secure Agent checks the status of the job completion. Doesn't apply to SQL warehouse. Default is 30 seconds.
DB REST API Timeout[1]	The Maximum time in seconds for which the Secure Agent retries the REST API calls to Databricks when there is an error due to network connection or if the REST endpoint returns `5xx HTTP` error code. Doesn't apply to SQL warehouse. Default is 10 minutes.
DB REST API Retry Interval[1]	The time Interval in seconds at which the Secure Agent must retry the REST API call, when there is an error due to network connection or when the REST endpoint returns `5xx HTTP` error code. Doesn't apply to SQL warehouse. This value does not apply to the Job status REST API. Use job status poll interval value for the Job status REST API. Default is 30 seconds.
Forward Rejected Rows	Determines whether the transformation passes rejected rows to the next transformation or drops rejected rows. By default, the agent forwards rejected rows to the next transformation.

[1]Doesn't apply to mappings in advanced mode.

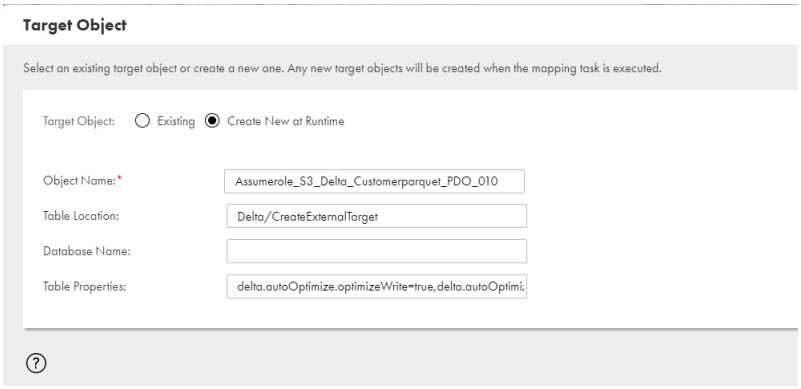
Create a target table at runtime

You can use an existing target or create a target to hold the results of a mapping. If you choose to create the target, the agent creates the target if it does not exist already when you run the task.

You can create both managed and external tables for mappings and mappings in advance mode.

To specify the target properties, perform the following tasks:

1. Select the Target transformation in the mapping.
2. To specify the target, click the **Target** tab.
3. Select the target connection.
4. For the target type, choose **Single Object** or **Parameter**.
5. Specify the target object or parameter.
6. To specify a target object, perform the following tasks:
   a. Click **Select** and choose a target object. You can select an existing target object or create a new target object at runtime.

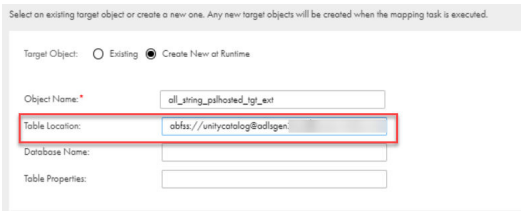


   b. To create a target object at runtime, select **Create New at Runtime**.
   c. In the **Object Name** field, enter the name of the target table that you want to create. Specify the table name in lowercase.
   d. In the **Table Location** field, enter the location of the target table data.

      The table location is relative to the data bucket or data filesystem name specified in the connection. External table is created if you specify the table location.

      For unity catalog, specify a pre-existing location to create a new target at runtime.

      When you use personal staging location, specify a pre-existing table location with an absolute path to the directory to create a new target at runtime. The following image shows the table location for personal staging location.



e. In the **Database Name** field, specify the Databricks database name.

   The database name that you specify in the connection properties takes precedence.

   Note: If you do not specify the Database Name when you create a new target or in the Database connection attribute, the default database is used to create a new target irrespective of the database name specified in the SQL Warehouse JDBC URL.

f. Specify the **Table Properties** to optimize the table configuration settings. You can use table properties to tag tables with information that are not tracked by SQL queries. To see the list of table properties and table options, see the Databricks documentation.

g. Click **OK**.

Rules and guidelines for create target at runtime

When you configure a mapping with the **Create New at Runtime** option, consider the following rules:

- When a source object consists of Date data type and you use the default create target option in a mapping, the date data gets corrupted. To resolve this issue, navigate to **Edit Metadata** option in the **Target fields** of the target and change **Native Type** to Date.
- When you enable dynamic schema handling in a task and create target at runtime, you must provide the complete path of the target table in the Database Name. Ensure that the table name is in lowercase. For example, database_name/TABLE/table_name.
- When you configure a mapping to create a new target at runtime and the source data contains complex fields, the metadata fails to appear on the **Target Fields** tab.
- Hierarchical data types are supported only when you create a new target at runtime and not for existing targets.
- Only **Insert** operation is applicable for hierarchical data types in the Target transformation

Override the update operation

You can specify an update override to override the update query that the Secure Agent generates for the update operation.

When you configure an update override, the Secure Agent uses the query that you specify, stages the data in files, and then loads that data into a temporary table using the merge command. The data from the temporary table is then loaded to the Databricks target table. The syntax that you specify for the update query must be supported by Databricks.

Specify the update override query in the following format:

```
MERGE INTO <Target table name> AS A USING :TU AS B ON A.<Column1> = B.<Column1> WHEN
MATCHED THEN UPDATE SET A.<Column2> = B.<Column2>, A.<Column3>= B.<Column3> ...
A.<ColumnN>= B.<ColumnN>
```

where, *:TU* represents the incoming data source for the target transformation. The Secure Agent replaces *:TU* with a temporary table name while running the mapping and does not validate the update query.

Note: While specifying the name of the target table, you must provide a fully qualified name in the format: `<Catalog_Name>.<Database_Name>.<Target_TableName>`.

When you configure an update override in a mapping to write to Databricks, consider the following rules and guidelines:

- Ensure that the column names for :TU matches the target table column names.
- Specify the update query with a valid SQL syntax because Databricks Connector replaces :TU with a temporary table name and does not validate the update query.

- Do not change the order of the column in the mappings when you configure the update override option.
- The column names used in the override query must not be unconnected in the target field mapping.
- You can only override a single SQL query at runtime.
- You cannot perform an update override on insert, upsert, delete, data driven, and dynamic schema handling operations.

Passthrough partitioning

You can configure partitioning to optimize the mapping or performance at run time when you write data to Databricks targets.

The partition type controls how the agent distributes data among partitions at partition points. You can define the partition type as passthrough partitioning. The agent distributes rows of target data based on the number of threads that you define for the partition.

Determine the order of processing for multiple targets

When you configure a mapping to write to multiple targets in a single pipeline, you can configure each target for any of the write operations. The order of the target operation is not deterministic.

However, if you want to process the target operations to process in a specific order such as delete, update, and insert, you need to set certain properties either in the Secure Agent or in the task properties.

Set -DEnableSingleCommit=true in the Secure Agent properties

Perform the following tasks to set the property for the Secure Agent:

1. Open Administrator and select **Runtime Environments**.
2. Select the Secure Agent for which you want to set the property.
3. On the upper-right corner of the page, click **Edit**.
4. In the **System Configuration Details** section, select the **Type** as **DTM** for the Data Integration Service.
5. Edit the JVM options and set the property to *-DEnableSingleCommit=true*.



Set the EnableSingleCommit property in the task properties

Perform the following tasks to set the property in the task:

1. On the **Runtime Options** tab in the mapping task properties, navigate to the Advanced Session Properties section.
2. From the **Session Property Name** list, select **Custom Properties**, and set the Session Property Value to **Yes**.



Optimize the staging performance of a mapping

Data Integration, by default, creates a flat file locally in a temporary folder to stage the data before writing to Databricks. You can set Data Integration to optimize the staging performance of the write operation on a Linux machine.

If you do not set the staging property, Data Integration performs staging without the optimized settings, which might impact the performance of the task.

You can optimize the staging performance only when you use the SQL warehouse to run the mappings.

Setting the staging property

You can optimize the mapping performance of the write operation on Linux.

You need to set the following staging property in the agent properties:

```
INFA_DTM_STAGING_ENABLED_CONNECTORS
```

Perform the following tasks to set the staging property for the Tomcat in the Secure Agent properties:

1. In Administrator, click **Runtime Environments**.
2. Edit the Secure Agent for which you want to set the property.
3. In the **System Configuration Details** section, select the **Service** as **Data Integration Server** and the type as **Tomcat**.
4. Set the value of the Tomcat property to the plugin ID of Databricks Connector.
   You can find the plugin ID in the manifest file located in the following directory: `<Secure Agent installation directory>/downloads/<Databricks Delta package>/CCIManifest`

   The following image shows the property set for the Secure Agent for the write operation:



When you run the mapping, a flat file is created in the following directory in your Linux machine:

```
\tmp\DELTA_stage
```

You can check the session logs. If the staging is done through the flat file successfully, Data Integration logs the following message in the session log:

```
The INFA_DTM_STAGING is successfully enabled to use the flat file to create local staging files.
```

Rules and guidelines

Consider the following guidelines when you enable the staging property for the write operation:

- If a mapping enabled for SQL ELT optimization runs without SQL ELT optimization because of any issues, Data Integration performs staging without the optimized settings even though you set the staging property to optimize the staging performance.
- When you read binary data and the data size exceeds 50 MB in a single cell, you need to adjust the default DTM staging file size of 50 MB set in the advanced target properties to a higher value proportional to your data size. However, ensure that the data size within a single cell does not exceed 70 MB to prevent data truncation in the target.
- When you write values of double data type to a Databricks target, the values are rounded off. To write exact values from the source to the target, use the decimal data type.

Lookups for Databricks

Add a Lookup transformation to retrieve data based on a specified lookup condition. When you add a Lookup transformation to a mapping, you define the lookup connection, lookup objects, and lookup properties related to Databricks.

In the Lookup transformation, select the lookup connection and object. Then, define the lookup condition and the outcome for multiple matches.
The mapping queries the lookup source based on the lookup fields and the defined lookup condition. The lookup operation returns the result to the Lookup transformation, which then passes the results downstream.

You can configure the following lookups:

- **Connected.** You can use a cached or uncached connected lookup for mappings. You can also use a dynamic lookup cache to keep the lookup cache synchronized with the target.
- **Unconnected.** You can use a cached lookup. You need to supply input values for an unconnected Lookup transformation from a :LKP expression in a transformation that uses an Expression transformation.

Lookup transformation doesn't apply to Databricks cluster.

Lookup objects are not supported for mappings in advanced mode.

For more information about Lookup transformation, see *Transformations* in the Data Integration documentation.

Lookup properties for Databricks

The following table describes the Databricks lookup object properties that you can configure in a Lookup transformation:

Property	Description
Connection	Name of the lookup connection. You can select an existing connection, create a new connection, or define parameter values for the lookup connection property. If you want to overwrite the lookup connection properties at runtime, select the **Allow parameter to be overridden at run time** option.
Source Type	Type of the source object. Select Single Object, Query, or Parameter.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the lookup object or click **New Parameter** to define a new parameter for the lookup object. The **Parameter** property appears only if you select parameter as the lookup type. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option.

Property	Description
Lookup Object	Name of the lookup object for the mapping.
Multiple Matches	Behavior when the lookup condition returns multiple matches. You can return all rows, any row, the first row, the last row, or an error. You can select from the following options in the lookup object properties to determine the behavior: - Return first row - Return last row - Return any row - Return all rows - Report error

The following table describes the Databricks lookup object advanced properties that you can configure in a Lookup transformation:

Advanced Property	Description
Database Name	Overrides the database specified in the connection.
Table Name	Overrides the table specified in the connection.
Staging Location	Relative directory path to store the staging files. - If the Databricks cluster is deployed on AWS, use the path relative to the Amazon S3 staging bucket. - If the Databricks cluster is deployed on Azure, use the path relative to the Azure Data Lake Store Gen2 staging filesystem name.
SQL Qverride	Overrides the default SQL query used to read data from the Databricks custom query source.
Pre-SQL	SQL statement that you want to run before reading data from the source.
Post-SQL	SQL statement that you want to run after reading data from the source.
Job Timeout	Maximum time in seconds that is taken by the Spark job to complete processing. If the job is not completed within the time specified, the Databricks cluster terminates the job and the mapping fails. If the job timeout is not specified, the mapping shows success or failure based on the job completion.
Job Status Poll Interval	Poll interval in seconds at which the Secure Agent checks the status of the job completion. Default is 30 seconds.
DB REST API Timeout	The Maximum time in seconds for which the Secure Agent retries the REST API calls to Databricks when there is an error due to network connection or if the REST endpoint returns `5xx HTTP` error code. Default is 10 minutes.
DB REST API Retry Interval	The time Interval in seconds at which the Secure Agent must retry the REST API call, when there is an error due to network connection or when the REST endpoint returns `5xx HTTP` error code. This value does not apply to the Job status REST API. Use job status poll interval value for the Job status REST API. Default is 30 seconds.

Advanced Property	Description
Tracing Level	Sets the amount of detail that appears in the log file. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Lookup Data Filter[1]	Limits the number of lookups that the mapping performs on the cache of the lookup source table based on the value you specify in the filter condition. This property is applicable when you enable lookup cache on the **Advanced** tab for the Lookup transformation. Maximum length is 32768 characters. For more information about this property, see *Transformations* in the Data Integration documentation.
[1]Doesn't apply to mappings in advanced mode.	

Multiple match options in lookups

When you configure a lookup, you define the behavior when a lookup condition returns more than one match. You can return all rows, any row, the first row, the last row, or an error.

The following configurations have multiple match policy restrictions:

- You cannot parameterize an uncached connected lookup with parameterized prefixes at source.
- When the data is not matched for an uncached connected lookup, incorrect data is logged in the target.

Enable lookup caching

When you configure a Lookup transformation in a mapping, you can cache the lookup data during the runtime session.

When you select **Lookup Caching Enabled**, Data Integration queries the lookup source once and caches the values for use during the session, which can improve performance. You can configure dynamic lookup caching and can specify the directory to store the cached lookup.

Note: You cannot perform insert and update operations to the same target at runtime when you dynamically cache the lookup objects in a mapping.

For information about lookup caching, see the chapter " Lookup transformations" in *Transformations* in the Data Integration documentation.

SQL transformation

You can configure an SQL transformation in a Databricks mapping to process SQL queries. The SQL transformation needs to be connected to the mapping pipeline.

When you add an SQL transformation to the mapping, on the **SQL** tab, you define the database connection and select the SQL query type for the transformation. The SQL transformation process the entered query that you define in the SQL editor. You can specify functions with a simple SELECT statement. Do not use more than one SQL query in an SQL transformation.

When you configure a SELECT query, specify the column names in the output ports for the functions. For example, when you specify the query `SELECT square(~AGE~), sqrt(~SNAME~)`, specify two output columns for the AGE and SNAME functions.

When you run the mapping, the SQL transformation processes the query and returns the rows. The SQL transformation also returns any errors that occur from the underlying database or from a user syntax that is not valid. When an SQL error occurs, the error is logged to the `SQLError` field, by default.

Note: SQL transformation doesn't apply to Databricks cluster.

For more information about SQL queries, see *Transformations* in the Data Integration documentation.

Rules and guidelines for SQL transformation

- You cannot configure a stored procedure in an SQL transformation.
- The SQL query must be a simple SELECT statement without FROM and WHERE arguments.
- You cannot configure a parameterized query in an SQL transformation.
- You cannot include special characters in the query.
- You need to manually specify the output ports, including the numrowsAffected and passthrough ports that are used downstream in the transformation.

Dynamic schema handling

You can choose how Data Integration handles changes that you make to the data object schemas. To refresh the schema every time the mapping task runs, you can enable dynamic schema handling in the task.

Configure schema change handling on the **Runtime Options** page when you configure the task.

The following table describes the schema change handling options:

Option	Description
Asynchronous	Default. Data Integration refreshes the schema when you edit the mapping or mapping task, and when Informatica Intelligent Cloud Services is upgraded.
Dynamic	Data Integration refreshes the schema every time the task runs. You can choose from the following options to refresh the schema: - **Alter and apply changes.** Data Integration applies the following changes from the source schema to the target schema: - New fields. Alters the target schema and adds the new fields from the source. - **Don't apply DDL changes.** Data Integration does not apply the schema changes to the target. - **Drop current and recreate.** Drops the existing target table and then recreates the target table at runtime using all the incoming metadata fields from the source.

Note: You cannot enable dynamic schema handling in a task to run a mapping in advanced mode. Dynamic schema handling doesn't apply to Databricks cluster.

For more information, see the "Schema change handling" topic in *Tasks* in the Data Integration help.

Parameterization

You can parameterize the following properties when you create mappings:

- Source properties. Source type, source connection, query options in source, database name, table name, and advanced properties in the source.
- Target properties. Target type, target connection, target database name, target table name, and target advanced properties.
- Lookup properties. Lookup type, lookup connection, lookup advanced properties.

You can parameterize the following properties when you create mappings in advanced mode:

- Source properties. Source type, source connection, database name, and table name in the source.
- Target properties. Target type, target connection, target database name, and target table name.

IDENTITY columns

You can use the IDENTITY or generated columns in mappings and mapping tasks.

When you write to a table with IDENTITY or generated columns, the values in the columns are automatically generated based on a user-specified function in the Databricks table.

You can use the IDENTITY column for insert, update, upsert, and delete operations. The query can be used for connected and unconnected ports for matching the columns based on the query in the CREATE TABLE statement.

The following table shows the behavior of the operations for each type of generated column:

Type of query	INSERT [INSERT INTO]	UPDATE [Merge]	UPSERT [Merge]	DELETE [Merge]
GENERATED ALWAYS AS	Port needs to be unconnected	Can be used for matching	Port needs to be unconnected and cannot be used for matching	Can be used for matching
GENERATED ALWAYS AS IDENTITY	Port needs to be unconnected	Can be used for matching	Port needs to be unconnected and cannot be used for matching	Can be used for matching
GENRATED BY DEFAULT AS IDENTITY	Port can be connected or unconnected	Can be used for matching	Port can be connected and can be used for matching	Can be used for matching

Note: You cannot define an IDENTITY column when you create a new target at runtime and the source table has generated columns. IDENTITY columns don't apply to Databricks cluster.

Mappings in advanced mode example

You work for a retail company that offers more than 50,000 products and the stores are distributed across the globe. The company ingests a large amount of customer engagement details from the transactional CRM system into Amazon S3.

The sales team wants to improve customer engagement and satisfaction at every touch point. To create a seamless customer experience and deliver personalized service across the various outlets, the retail company plans to load the data that is stored in the Amazon S3 bucket to Databricks.

You can create a mapping that runs on an advanced cluster to achieve faster performance when you read data from the Amazon S3 bucket and write data to the Databricks target.

You can choose to add transformations to process the raw data that you read from the Amazon S3 bucket and then write the curated data to Databricks.

The following example illustrates how to create a mapping in advanced mode to read from an Amazon S3 source and write to Databricks target:

1. In Data Integration, click **New** > **Mappings** > **Mapping**.
2. In the Mapping Designer, click **Switch to Advanced**.

   The Mapping Designer updates the mapping canvas to display the transformations and functions that are available in advanced mode.
3. Enter a name, location, and description for the mapping.
4. Add a Source transformation, and specify a name and description in the general properties.
5. On the **Source** tab, perform the following steps to read data from the Amazon S3 source:
   a. In the **Connection** field, select the Amazon S3 V2 connection.
   b. In the **Source Type** field, select single object as the source type.
   c. In the **Object** field, select the parquet file object that contains the customer details.
   d. In the **Advanced Properties** section, specify the required parameters.
6. On the **Expression** tab, define an expression to change the file name port of the customer parquet file to uppercase based on your business requirement before you write data to the Databricks target.
7. Add a Target transformation, and specify a name and description in the general properties.
8. On the **Target** tab, specify the details to write data to Databricks:
   a. In the **Connection** field, select the Databricks target connection.
   b. In the **Target Type** field, select single object.
   c. In the **Object** field, select the Databricks object to which you want to write the curated customer engagement data.
   d. In the **Operation** field, select the insert operation.
   e. In the **Advanced Properties** section, specify the required advanced target properties.
9. Click **Save** > **Run** to validate the mapping.

   In Monitor, you can monitor the status of the logs after you run the task.

Rules and guidelines for mappings

Consider the following rules and guidelines for Databricks objects used as sources, targets, and lookups in mappings:

- If you run a mapping on the Databricks cluster to write to a Databricks target and the Timestamp data in the source contains the date 01-01-0001, the date is incorrectly written to the target.
- Dates earlier than 1582-10-15 in the date data type are incorrectly written to the target.
- You cannot use input or in-out parameters in a parameter file to parameterize multiple objects or parameterize the relationships between the objects.
- When you use a parameter file to parameterize the table name, specify only the table name in the parameter value. Do not specify the database name with the table name in the parameter value. For example, `<databasename>.<tablename>` or `<databasename>/<tablename>`.
- Ensure that column names do not contain unicode characters.
- A mapping with Null values in an uncached lookup condition generates incorrect results.
- When you do not specify the database name in Databricks connection and read multiple objects with the same table name from different databases, you must append the database name for each object in the advanced relationship.
- When you specify SESSSTARTTIME variable in a query in a mapping task to return the Datetime values, specify the query in the following format:
  `:select to_timestamp('$$$SESSSTARTTIME', 'MM/dd/yyyy HH:mm:ss.SSSSSS') as t;`
- When you run multiple concurrent mappings to write data to Databricks targets, a transaction commit conflict error might occur and the mappings might fail.
- View objects are displayed in the Table panel instead of the View panel while importing a Databricks object. This issue occurs when the Databricks cluster is deployed on AWS cloud service.
- To avoid java heap space error when you read or write complex files, set the JVM options for type DTM to increase the -Xms and -Xmx values in the system configuration details of the Secure Agent. The recommended values for -Xms is 512 MB and -Xmx is 1024 MB.
- When you import views, the Select Source Object dialog box does not display view objects.
- When you test the Databricks connection, the Secure Agent does not validate the values you specify in the Org ID connection parameter.
- You cannot use the Hosted Agent as a runtime environment when you configure a mapping to run on the SQL warehouse to read or write data that contains unicode characters.
- The number of clusters that the Secure Agent creates to run the mapping depends on the number of Databricks connections used in the transformations in a mapping. For example, if multiple transformations use the same Databricks connection, the mapping runs on a single cluster.
- When you keep the mapping designer idle for more than 15 minutes, the metadata fetch throws an exception.
- If you change the database name in the connection and run the existing mappings, the mappings start failing. After you change the database name in the connection, you must re-import the objects in the existing mappings before you run the mappings.
- Use the following formats to run the mapping successfully, when you import a Databricks source object containing Date or Boolean data types with a simple source filter conditions:

  - Boolean = 0 or 1

  - Date = YYYY-MM-DD HH24:MM:SS.US

- When you run a mapping with source column data type as string containing TRUE / FALSE value and write data to target with Boolean data type column of a Databricks table, the Secure Agent writes data as 0 to the target.
- When the Databricks analytics cluster is down and you perform a test connection or import an object, the connection is timed out after 10 minutes.
- When you parameterize the source or target connection in a mapping and you do not specify the database name, ensure that you specify the database name in lowercase when you assign a default value for the parameter.
- When you parameterize the source filter condition or any expressions in a mapping, ensure that you specify the table name in lowercase when you add the source filter condition or the expression in the mapping task. Otherwise, the Secure Agent throws the following exception:
  `Invalid expression string for filter condition`
- When you run a mapping to write data to a Databricks target using create target at runtime and the target table already exists, ensure that the target table schema is same. Otherwise, the mapping fails.
- When you run a mapping to write data to multiple Databricks targets that use the same Databricks connection and the Secure Agent fails to write data to one of targets, the mapping fails and the Secure Agent does not write data to the remaining targets.
- When you use the `Create New at Runtime` option to create a Databricks target, you can parameterize only the target connection and the table name using a parameter file. You cannot parameterize other properties such as `Path` or `DBname`.
- The pre-SQL and post-SQL commands run non-linearly. In the session logs, you will see that the target pre-SQL cases are executed before the source pre-SQL queries.
- When you run pre-SQL and post-SQL commands to read from sources that have semi-colons within the query, the mappings fails. The queries can only have semi-colons at the end.
- When you read or write Unicode data to Databricks on the SQL endpoint, you need to set some properties for the Secure Agent before you run the mapping. You can perform one of the following tasks:
  - Set the following environment variables in the Secure Agent machine, and then restart the Secure Agent.

    export LANGUAGE="en_US.UTF-8"

    export LC_ALL="en_US.UTF-8"
  - Configure the property `-Dfile.encoding=UTF-8` in the JVM options in the Secure Agent properties.

  Note: You cannot read or write Unicode data when you use the Hosted Agent in the connection.
- When you configure a mapping where you have staged data in the Personal Staging Location, the temporary data is not deleted when the mapping stops abruptly.
- When you parameterize the source or lookup object, ensure that the column names in the source object and lookup object are not the same. Else, the mapping fails.

Rules and guidelines for mappings in advanced mode

Consider the following rules and guidelines for Databricks objects used as sources and targets in mappings in advanced mode:

- When you write data to multiple Databricks targets with the same table and configure different target operations for each target, the mapping throws a concurrent append exception.

- When mappings in advanced mode reads NULL values from the source and updates or upserts a column with NOT NULL constraint in the Databricks target table, the mapping fails and the Secure Agent fails to log an appropriate error message.
- When you read data from or write data to Databricks and the source or target object contains 5000 or more columns, the mapping fails.
- A mapping in advanced mode configured to read from or write to Databricks fails in the following cases:

  - Data is of the Date data type and the date is less than 1582-10-15.

  - Data is of the Timestamp data type and the timestamp is less than 1900-01-01T00:00:00Z.

  To resolve this issue, specify the following spark session properties in the mapping task or in the custom properties file for the Secure Agent:

  - `spark.sql.legacy.timeParserPolicy=LEGACY`

  - `spark.sql.parquet.int96RebaseModeInWrite=LEGACY`

  - `spark.sql.parquet.datetimeRebaseModeInWrite=LEGACY`

  - `spark.sql.parquet.int96RebaseModeInRead=LEGACY`

  - `spark.sql.parquet.datetimeRebaseModeInRead=LEGACY`

  - `spark.sql.avro.datetimeRebaseModeInWrite=LEGACY`

  - `spark.sql.avro.datetimeRebaseModeInRead=LEGACY`
- When you do a data type conversion from Float to Double or use create target at run time, data loss is encountered.
- If a mapping in advanced mode has a source column data type as String containing true or false value and a target data type as Boolean, the Secure Agent writes data as null to the target.
- Use the following formats when you import a Databricks source object containing Boolean, Date, or Timestamp data types with a simple source filter conditions:

  - Boolean = 0 or 1

  - Date = YYYY-MM-DD HH24:MM:SS.US

  - Timestamp = YYYY-MM-DD HH24:MM:SS.US
- You cannot use the following features:

  - View

  - Multipipe
- After you create and run a mapping configuration task, it is recommended to shut down the job cluster. If you modify a mapping task or edit the connection linked to a mapping task, metadata is fetched again and the job cluster restarts.
- When you do a data type conversion from Date or Timestamp to String, the Secure Agent writes the value only in the following default format for both Date and Timestamp:
  `MM/DD/YYYY HH24:MI:SS`
- When you do a data type conversion from String to Date or Timestamp, the String value must be in the following format:
  `MM/DD/YYYY HH24:MI:SS`

  To use any other format, you must specify the format in the advanced session property of a mapping task for successful conversion. Null is populated in the target for the unmatched format.
- When you do a data type conversion from Bigint to Double, the target data is written in the exponential format.

- When you perform an update, upsert, or a data driven operation with an IIF condition that includes DD_DELETE or DD_UPDATE, ensure that the update column that you specified does not have duplicate rows. Otherwise, the mapping fails with the following error:
  `java.lang.UnsupportedOperationException: Cannot perform MERGE as multiple source rows matched and attempted to update the same target row in the Delta table.`
- When you perform an insert, update, upsert operation, or DD_UPDATE and the range of the data in source column is greater than the range of the target column, the mapping does not fail and leads to data truncation.
- When you specify a single constant in the data driven condition, the mapping ignores the data driven condition and the Secure Agent performs insert, update, or delete operation based on the constant. For example, if you specify the data driven condition as DD_INSERT, the mapping does not consider the update columns and depends on the Write Disposition property.
- When you specify a single constant with the IIF condition in the data driven condition such as IIF(COL_INT > 20 , DD_UPDATE), the Secure Agent inserts the data into the target even for those rows that do not satisfy the condition.
- When you specify the DD_REJECT constant in the data driven condition, the Secure Agent does not log the rejected rows in the error file or the session log.
- You can run mappings with hierarchical data types only on a Linux system.
- When you configure mappings with nested statements that contain hierarchical data types, the mapping fails if the nested field names contain the following special characters:

  - ,

  - ( )

  - < >
- When you read and write data of hierarchical data type, ensure that the column names do not start with a number.
- When you run a Databricks mapping in advanced mode to write data to multiple Databricks targets and enable SQL ELT optimization, the mapping runs successfully but the data is written to only one target.
- When you read data of hierarchical data type, ensure that the column names do not contain special characters.

CHAPTER 4

Migrating a mapping

You can configure a connection and mapping in one environment and then migrate and run the mapping in another environment.

You can also migrate mappings configured in advanced mode. After the migration, you can change the connection properties from the Administrator service, but you do not need to modify the mapping. Data Integration uses the configured runtime attributes from the earlier environment to run the mapping successfully in the new environment.

Plan the migration

When you run the migrated mapping in the new environment, you might want to use the same or a different connection endpoint or object path. Based on your requirement, follow the guidelines in this section before you migrate a mapping.

The following table lists the configured advanced properties for an object type that you can retain when you migrate a mapping:

Object Type	Advanced properties
Single object	Database, table
Query	SQL override

Migrate a mapping within the same path

If you want the migrated mapping to use the same object path as in the earlier environment, you must maintain the same database and table in the Databricks account for both the earlier and the new environment.

For example, if you have two different accounts, Account1 used for Organization 1 and Account2 used for Organization 2, the object path for the database and table must be the same in both the accounts:

Account1: DB1/TABLE1

Account2: DB1/TABLE1

You do not need to override the database and table in the advanced properties.

Migrate a mapping to a different path

You can use a different object path to run the mapping from the new environment.

Before you migrate the mapping, you can change the object metadata, runtime attributes, or the connection attributes to reflect the object path in the migrated environment. You do not have to edit or update the mapping in the new environment.

As a rule, when you specify the database or table in the advanced properties, connection, or object properties, Data Integration honors the attributes in the following order of precedence:

1. **Runtime advanced attributes**. The advanced properties such as database and table in the Source, Target, or Lookup transformation in a mapping.
2. **Connection attributes**. Attributes such as catalog and database set in the connection properties.
3. **Object metadata**. The object selected in the Source, Target, or Lookup transformation in a mapping.

Migration options

When you migrate to a different path, you can choose from one of the following options to update the object path:

Option 1. Update the connection properties to reference the new object

When you import the mapping from, for example, Org 1 into a new organization, for example, Org 2, you can change the existing connection in the **Review Connections** section to map to the connection that has access to the specified database and table in Org 2.

Option 2. Override the properties from the advanced properties

Before the migration, specify the required database and table for the object in the new organization in the advanced properties of the Org 1 mapping.

After the migration, when you run the mapping, the Secure Agent uses the configured advanced parameters to override the object specified in the mapping imported from Org 1.

Option 3. Parameterize the properties in the mapping

You can choose to parameterize the advanced attributes, such as the database and table before the migration. You can configure input parameters, in-out parameters, and parameter files in the mapping. When you use a parameter file, you can save the parameter file on a local machine or in a cloud-hosted directory. After you migrate the mapping, do not edit or update the mapping. If you have used in-out parameters for the advanced attributes such as for the database and table, you can update these from the parameter file.

Parameterizing only the advanced properties, but not the object

If you want to parameterize only the advanced properties and use them at runtime, select a placeholder object in the object properties in the mapping and then specify an override to this placeholder object from the advanced properties. Ensure that the placeholder object contains the same metadata as the corresponding table that you specify as an override. When you run the mapping, the value specified in the advanced property overrides the placeholder object.

Parameterizing both the object and the advanced properties

If you want to keep both the Databricks object type and the advanced fields parameterized, you must leave the **Allow parameter to be overridden at runtime** option unselected in the input parameter window while adding the parameters, and then select the required object at the task level. When you run the task, the values specified in the advanced properties take precedence.

Parameterization is not applicable for mappings that use the **Create Target** option. In advanced mode, only input parameters are applicable for migration.

Migration restrictions

You need to consider some rules for some of the configurations before you migrate assets.

Advanced filter and table name override combination

If the mapping contains an advanced filter for the object, for example, <Table2>, in the query options section and an override to the table name in the advanced properties, consider the following rules to override the table:

In mappings, consider one of the following options:

- The advanced filter in the mapping must include the `$$Filtercondition` value. In the mapping task, you can override the value from the parameter file in the task properties. The parameter condition in the parameter file must include the following condition: `$$Filtercondition =<Table2>.id >= 1`
- Use the table name that you specify as an override in the advanced properties directly in the advanced filter condition. For example, `<Table2>.id >= 1`

When you create mappings in advanced mode, use the table name specified as an override in the advanced properties directly in the advanced filter condition. For example, `<Table2>.id >= 1`

Parameterization

Consider the following rules to parameterize the object and advanced properties:

- Parameterization is not applicable when you create a new target at runtime and select the Allow parameter to be overridden at runtime option in a mapping.
- Parameterization is not applicable for mappings in advanced mode in the migration use case.
- If there are multiple pipelines configured in a mapping, do not parameterize the Databricks object. You must select a placeholder object while creating the mapping before you migrate.

Schema change handling

You cannot dynamically refresh the data object schema at runtime. You must maintain the same metadata for the table selected in the source, target, or lookup transformations and the corresponding advanced field overrides, as schema change handling is not applicable.

SQL override and custom query combination

In the mapping, when you specify an SQL override and the object type used is a custom query, ensure that the table you specified in the custom query and the SQL override contains the same metadata. The custom query and SQL override must be fully qualified. For example, `Select * from Catalog1.DB1.Table1`

When the mapping contains an SQL override and custom query combination, do not specify the database or table name in both the advanced source properties in the mapping and in the Databricks connection.

When you configure a partially parameterized custom query, specify the default value for the parameters. For example, when you specify `Select * from $$Catalog1.$$DB1.$$Table1`, add a placeholder value as the default value while creating the in-out parameters for Catalog1, DB1, and Table1.

If you have selected parameter as the source type in the mapping, do not select the **Allow parameter to be overridden at runtime** option. You can then select the custom query in the mapping task.

Table name override

In the mapping, if you select the custom query object type and specify a database override or table name override, Data Integration ignores the override values and does not override the database or table.

Update override

When you write data to Databricks and specify an update override for the target object, after you migrate the mapping, ensure that the target object specified in the override exists in the mapping.

CHAPTER 5

SQL ELT with Databricks Connector

You can enhance the mapping performance with SQL ELT.

You can use SQL ELT to read data from a cloud data warehouse and write to the same cloud data warehouse. You can also read data from a data lake in your cloud ecosystem and write it to a cloud data warehouse in the same ecosystem. Data Integration translates the transformation logic into ecosystem-specific commands and SQL statements that run in the underlying cloud infrastructure. This increases the data processing speed because the data isn't moved out of the cloud infrastructure for processing.

Example

You work in the sales and inventory department of a retail organization that operates multiple stores and an online platform. The management wants to optimize its inventory management, understand customer behavior, and improve sales strategies by analyzing data from various sources.

The organization uses Amazon S3 to store the data from multiple sources. To analyze the data, the organization plans to leverage the warehouse infrastructure of Databricks and load all its data to Databricks.

You can use SQL ELT to quickly load data from Amazon S3 to Databricks. SQL ELT is faster than the ETL method and eliminates any data egress costs as the data remains within your cloud infrastructure.

SQL ELT configuration options

You can either use the Mapping - SQL ELT mapping type to directly create a mapping in SQL ELT mode, or you can first create a mapping and then enable SQL ELT optimization from the mapping task properties.

You can only use SQL ELT when you connect to a SQL warehouse to read from and write to Databricks tables.

Create a mapping in SQL ELT mode

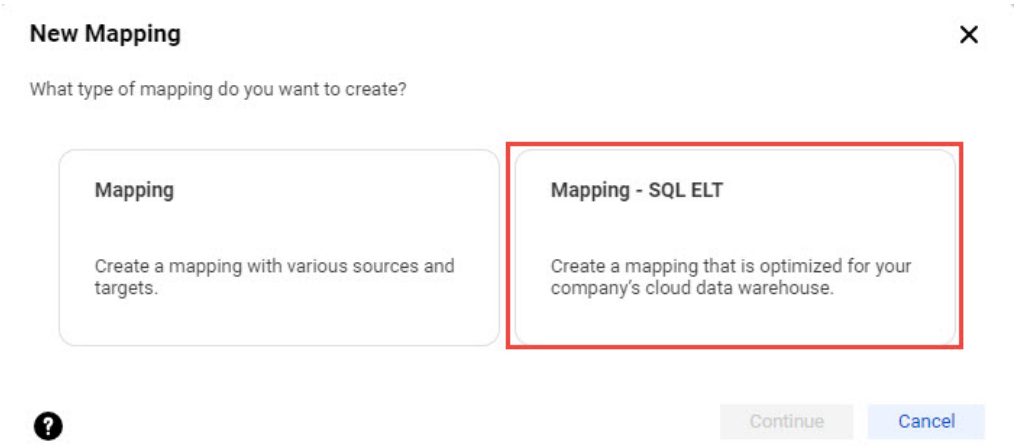
In a mapping in SQL ELT mode, you don't have to specify an optimization type. Data Integration pushes the transformation logic by default to the target database.

You can run a mapping in SQL ELT mode to load data from the following data sources to Databricks:

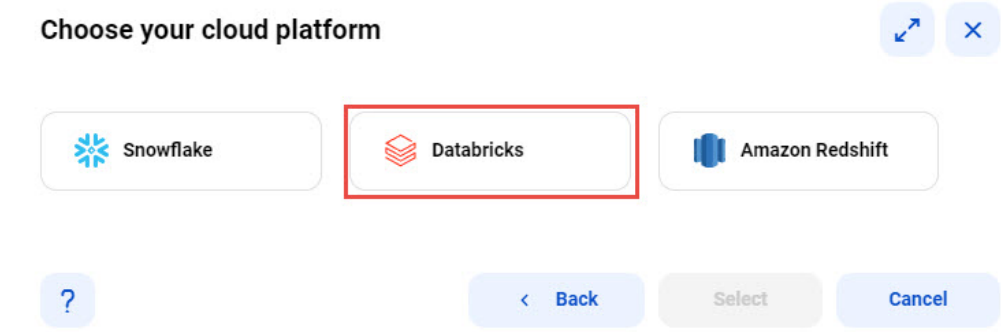
- Amazon S3
- Databricks

- Microsoft Azure Data Lake Storage Gen2

1. To create a mapping in SQL ELT mode, you create a mapping and select **Mapping - SQL ELT** as the mapping type.

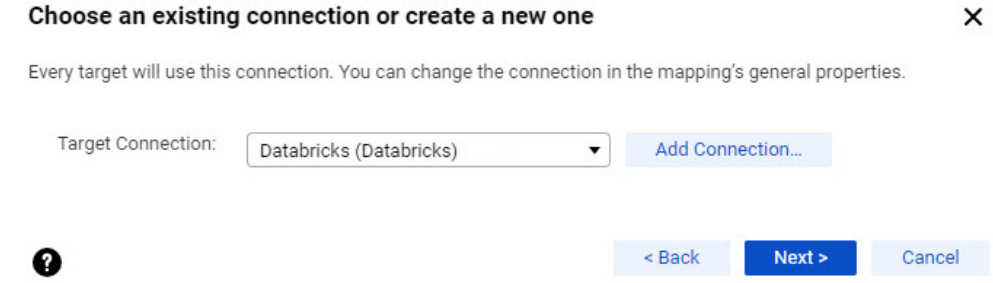


2. Select the cloud ecosystem where you want all the mapping logic to be processed.



3. You're then prompted to select the target connection.

Figure 1.



The mapping is automatically configured with the target connection you chose.

You can add additional transformations to the mapping. The transformations available in the transformation palette are transformations that the target endpoint can process. You can also preview data for individual transformations to validate the mapping logic.

For more information on mappings in SQL ELT mode, see Mappings in SQL ELT mode for Databrickson page 55.

Enable SQL ELT optimization for a mapping task

In a task enabled with SQL ELT optimization, you can select the optimization type to push some or all the transformation logic. Data Integration pushes the transformation logic to the source or target database based on the optimization type you specify in the task properties.

To enable SQL ELT optimization, add the configured mapping or mapping in advanced mode to a mapping task, and then enable SQL ELT optimization in the mapping task.

You can configure Full, Source, and other optimization configurations in the **SQL ELT optimization** advanced session properties on the **Runtime Options** tab. Data Integration processes any transformation logic that isn't pushed to the sources and targets on the on the Data Integration Server and on an advanced cluster.

You can enable SQL ELT optimization in a mapping task to load data from the following data sources to Databricks:

- Amazon S3
- Databricks
- Microsoft Azure Data Lake Storage Gen2

For more information on SQL ELT optimization, see SQL ELT optimization for a mapping taskon page 63.

SQL ELT query preview

Before you run a mapping that is configured for SQL ELT optimization or a mapping in SQL ELT mode, you can preview SQL queries in the SQL ELT Query panel in the Mapping Designer.

After you configure a mapping and run the preview, Data Integration creates and runs a temporary SQL ELT preview mapping task. When the job completes, Data Integration displays the SQL queries to be executed and any warnings in the **SQL ELT Query** panel. The warning messages help you understand which transformations in the configured mapping are not applicable for SQL ELT optimization. If SQL ELT query preview fails, Data Integration lists any queries generated up to the point of failure. You can edit the mapping and fix the required transformations before you run the mapping for SQL ELT optimization.

You can also view the temporary job created under **My Jobs** and download the session log to view the queries generated.

For more information about how to preview SQL ELT query, see "SQL ELT query preview" in *Mappings* in the Data Integration documentation.

Mappings in SQL ELT mode for Databricks

You can create mappings in SQL ELT mode to read data from Databricks, Amazon S3, or Microsoft Azure Data Lake Storage Gen2, load it to Databricks, and perform all of the data transformation within Databricks.

To create a mapping in SQL ELT mode, you create a mapping, and then select **Mapping - SQL ELT** as the mapping type. You're then prompted to choose Databricks as the cloud ecosystem and a Databricks target connection. If your organization doesn't have any Databricks connections, you're prompted to create one.

After you choose the target connection, the Mapping Designer opens. When you create a mapping, a Source transformation and a Target transformation are already on the canvas for you to configure.

Sources in mappings in SQL ELT mode

When you configure the source connection in the Source transformation, you can choose an Amazon S3 V2, a Databricks, or a Microsoft Azure Data Lake Storage Gen2 connection.

Amazon S3 V2 source properties

You can configure the following properties for an Amazon S3 V2 source:

- Source connection - Parameter, Allow parameter to be overridden at run time
- Source Type - Single, Parameter
- Format - Delimited, Avro, JSON, ORC, and Parquet
- Source Type - File and directory
- Folder Path
- File Name

For information on how to configure the supported properties, see the Amazon S3 V2 Connector documentation.

Databricks source properties

You can configure the following properties for a Databricks source:

- Source connection - Parameter, Allow parameter to be overridden at run time
- Source Type - Single, query, parameter
- Database Name
- Table Name
- SQL Override

Microsoft Azure Data Lake Storage Gen2 source properties

You can configure the following properties for a Microsoft Azure Data Lake Storage Gen2 source:

- Source connection - Parameter, Allow parameter to be overridden at run time
- Source Type - Single, Parameter
- Format - Delimited, Avro, JSON, ORC, and Parquet
- Filesystem Name Override
- Source Type - File and directory
- Allow Wildcard Characters
- Directory Override
- File Name Override

For information on how to configure the supported properties, see the Microsoft Azure Data Lake Storage Gen2 Connector documentation.

Targets in mappings in SQL ELT mode

When you configure a Target transformation in a mapping in SQL ELT mode, you need to use only a Databricks connection.

You can configure the following properties in a Databricks target transformation in a mapping in SQL ELT mode:

- Target Type - Single Object, Parameter, Create New Target at Runtime

- Operation - Insert, Update, Delete, Data Driven
- Target Database Name
- Target Table Name
- Write Disposition - Append, Truncate
- Update Mode - Update as Update, Update else Insert

For information on how the targets in mappings in SQL ELT mode behaves differently from the targets in other types of mappings, see Targets in mappings in SQL ELT mode in *Mappings* in the Data Integration help.

Transformations in mappings in SQL ELT mode

A mapping in SQL ELT mode includes transformations that Databricks can process.

You can use the following transformations in a mapping in SQL ELT mode:

- Source
- Target
- Aggregator
- Expression
- Filter
- Joiner
- Lookup
- Rank
- Router
- Sorter
- Union

Functions in mappings in SQL ELT mode

When you create expressions within a mapping in SQL ELT mode, you must use the native functions and expression syntax of Databricks and not Informatica functions and expression syntax.

You can use the following native functions in a mapping in SQL ELT mode:

Aggregate functions

Function	Function	Function
ANY()	FIRST()	REGR_INTERCEPT()
ANY_VALUE()	FIRST_VALUE()	REGR_R2()
APPROX_COUNT_DISTINCT()	HLL_SKETCH_AGG()	REGR_SLOPE()
APPROX_PERCENTILE()	HLL_UNION_AGG()	REGR_SXX()
AVG()	KURTOSIS()	REGR_SXY()
BIT_AND()	LAST()	REGR_SYY()

Function	Function	Function
BIT_OR()	LAST_VALUE()	SKEWNESS()
BIT_XOR()	MAX()	SOME()
BITMAP_CONSTRUCT_AGG()	MAX_BY()	STD()
BITMAP_OR_AGG()	MEAN()	STDDEV()
BOOL_AND()	MEDIAN()	STDDEV_POP()
BOOL_OR()	MIN()	STDDEV_SAMP()
CORR()	MIN_BY()	SUM()
COUNT()	MODE()	TRY_AVG()
COUNT_IF()	PERCENTILE()	TRY_SUM()
COUNT_MIN_SKETCH()	PERCENTILE_APPROX(	VAR_POP()
COVAR_POP()	REGR_AVGX()	VAR_SAMP()
COVAR_SAMP()	REGR_AVGY()	VARIANCE()
EVERY()	REGR_COUNT()	-

String and binary functions

Function	Function	Function
AES_DECRYPT()	INSTR()	RPAD()
AES_ENCRYPT()	LCASE()	RTRIM()
ASCII()	LEFT()	SHA()
BASE64()	LEN()	SHA1()
BIN()	LENGTH()	SHA2()
BINARY()	LEVENSHTEIN()	SOUNDEX()
BIT_LENGTH()	LOCATE()	SPACE()
BITMAP_COUNT()	LOWER()	SPLIT_PART()
BTRIM()	LPAD()	STARTSWITH()
CHAR()	LTRIM()	STRING()
CHAR_LENGTH()	MASK()	SUBSTR()

Function	Function	Function
CHARACTER_LENGTH()	MD5()	SUBSTRING()
CHARINDEX()	OCTET_LENGTH()	SUBSTRING_INDEX()
CHR()	OVERLAY()	TO_BINARY()
CONCAT()	PARSE_URL()	TRANSLATE()
CONCAT_WS()	POSITION()	TRIM()
CONTAINS()	REGEXP_COUNT()	TRY_AES_DECRYPT()
CRC32()	REGEXP_EXTRACT()	TRY_TO_BINARY()
DECODE()	REGEXP_INSTR()	UCASE()
ENCODE()	REGEXP_REPLACE()	UNBASE64()
ENDSWITH()	REGEXP_SUBSTR()	UNHEX()
FIND_IN_SET()	REPEAT()	UPPER()
FORMAT_NUMBER()	REPLACE()	URL_DECODE()
HEX()	REVERSE()	URL_ENCODE()
INITCAP()	RIGHT()	-

Numeric functions

Function	Function	Function
ABS()	DOUBLE()	RAND()
ACOSH()	E()	RANDN()
ACOS()	EXP()	RANDOM()
ASIN()	EXPM1()	RINT()
ASINH()	FACTORIAL()	ROUND()
ATAN()	FLOAT()	SEC()
ATAN2()	FLOOR()	SHIFTLEFT()
ATANH()	GETBIT()	SHIFTRIGHT()
BIGINT()	HYPOT()	SHIFTRIGHTUNSIGNED()
BIT_COUNT()	INT()	SIGN()

Function	Function	Function
BIT_GET()	ISNAN()	SIGNUM()
BIT_REVERSE()	LN()	SIN()
BITMAP_BIT_POSITION()	LOG()	SINH()
BITMAP_BUCKET_NUMBER()	LOG10()	SMALLINT()
BROUND()	LOG1P()	SQRT()
CBRT()	LOG2()	TAN()
CEIL()	MOD()	TANH()
CEILING()	NANVL()	TINYINT()
CONV()	NEGATIVE()	TRY_ADD()
COS()	PI()	TRY_DIVIDE()
COSH()	PMOD()	TRY_MULTIPLY()
COT()	POSITIVE()	TRY_SUBTRACT()
CSC()	POW()	WIDTH_BUCKET()
DECIMAL()	POWER()	-
DEGREES()	RADIANS()	-

Date and time functions

Function	Function	Function
ADD_MONTHS()	DAYOFYEAR()	TIMESTAMP()
CURDATE()	FROM_UNIXTIME()	TIMESTAMP_SECONDS()
CURRENT_DATE()	FROM_UTC_TIMESTAMP()	TO_DATE()
CURRENT_TIMESTAMP()	GETDATE()	TO_TIMESTAMP()
CURRENT_TIMEZONE()	HOUR()	TO_UNIX_TIMESTAMP()
DATE_ADD()	LAST_DAY()	TO_UTC_TIMESTAMP()
DATE_FORMAT()	MAKE_DATE()	TRUNC()
DATE_FROM_UNIX_DATE()	MAKE_TIMESTAMP()	TRY_TO_TIMESTAMP()
DATE_PART()	MINUTE()	UNIX_DATE()

Function	Function	Function
DATE()	MONTH()	UNIX_MICROS()
DATE_SUB()	MONTHS_BETWEEN()	UNIX_MILLIS()
DATE_TRUNC()	NEXT_DAY()	UNIX_SECONDS()
DATEADD()	NOW()	UNIX_TIMESTAMP()
DATEDIFF()	QUARTER()	WEEKDAY()
DAY()	SECOND()	WEEKOFYEAR()
DAYOFMONTH()	TIMESTAMP_MICROS()	YEAR()
DAYOFWEEK()	TIMESTAMP_MILLIS()	-

Cast functions

Function	Function	Function
BOOLEAN()	TO_NUMBER()	TRY_TO_NUMBER()
TO_CHAR()	TO_VARCHAR()	-

Miscellaneous functions

Function	Function	Function
ASSERT_TRUE()	IFF()	NVL()
COALESCE()	IFNULL()	RAISE_ERROR()
CURRENT_CATALOG()	INPUT_FILE_BLOCK_LENGTH()	REFLECT()
CURRENT_DATABASE()	INPUT_FILE_BLOCK_START()	SECRET()
CURRENT_METASTORE()	INPUT_FILE_NAME()	SESSION_USER()
CURRENT_SCHEMA()	IS_ACCOUNT_GROUP_MEMBER()	SPARK_PARTITION_ID()
CURRENT_USER()	IS_MEMBER()	TRY_REFLECT()
ELT()	ISNOTNULL()	TYPEOF()
EQUAL_NULL()	ISNULL()	USER()
GREATEST()	JAVA_METHOD()	UUID()
HASH()	LEAST()	VERSION()
HLL_SKETCH_ESTIMATE()	LUHN_CHECK()	XXHASH64()

Function	Function	Function
HLL_UNION()	MONOTONICALLY_INCREASING_ID()	-
IF()	NULLIF()	-

For more information about the native functions and expression syntax of Databricks, see Databricks functions in Databricks documentation.

Operators in mappings in SQL ELT mode

When you use mappings in SQL ELT mode, Data Integration converts the expression in the transformation by determining equivalent operators in the database.

The table lists the operators that you can push to Databricks:

Operator	Operator	Operator
+	||	<=
-	>	!=
*	<	AND
/	=	OR
%	>=	NOT

Rules and guidelines in mappings in SQL ELT mode

Consider the following rules and guidelines when you run mappings in SQL ELT mode:

Functions

- When you use the following functions, ensure that all the arguments are of the same data type:

  COALESCE

  DECODE

  ELT

  GREATEST

  IF

  IFF

  IFNULL

  LEAST

  NVL

  NVL2

- When you use the following functions, the number of optional arguments must not exceed five:

  COALESCE

  CONCAT

  CONCAT_WS

  COUNT

  ELT

  GREATEST

  HASH

  JAVA_METHOD

  LEAST

  REFLECT

  TRY_REFLECT

  XXHASH64

  In the following example of CONCAT_WS function, the first argument `','` is mandatory and the other arguments are optional.

  ```
  concat_ws(',', 'This', 'is', 'a', 'Spark', 'SQL');
  ```

  The output of this function is `This,is,a,Spark,SQL`

SQL ELT optimization for mapping tasks

When you run a task configured for SQL ELT optimization, the task converts the transformation logic to an SQL query. The task sends the query to the database, and the database executes the query.

The amount of transformation logic that you can push to the database depends on the database, transformation logic, and task configuration. The Secure Agent processes all transformation logic that it cannot push to the database.

Configure SQL ELT optimization for a mapping in the tasks properties. You can configure SQL ELT optimization in task that references a mapping or a mapping in advanced mode.

Note: You can only configure SQL ELT optimization when you use the SQL warehouse to connect to Databricks.

SQL ELT optimization types

When you apply SQL ELT optimization, the task pushes transformation logic to the source or target database based on the SQL ELT optimization type you specify in the task properties. Data Integration translates the transformation logic into SQL queries or Databricks commands to the Databricks database. The database runs the SQL queries or Databricks commands to process the transformations.

You can configure the following SQL ELT optimization types in a mapping:

None

The task does not push down the transformation logic to the Databricks database.

Full

The task pushes as much of the transformation logic as possible to process in the Databricks target database.

Data Integration analyses all the transformations from the source to the target. If all the transformations are compatible in the target, it pushes the entire mapping logic to the target. If it cannot push the entire mapping logic to the target, Data Integration first pushes as much transformation logic to the source database and then pushes as much transformation logic as possible to the target database.

When a transformation is not supported in the mapping, the task partially pushes down the mapping logic to the point where the transformation is supported for SQL ELT optimization. However, this applies only to SQL warehouse.

When you enable full SQL ELT optimization, you can determine how Data Integration handles the job when SQL ELT optimization does not work in the **Fallback Option** menu.

Source

The task pushes as much as the transformation logic as possible to process in the Databricks source database. This applies only to SQL warehouse.

Note: You cannot enable source SQL ELT optimization for mappings in advanced mode.

Read from and write to Databricks

You can configure SQL ELT optimization in a mapping to read from and write to Databricks using a Databricks connection.

Example

You work in a motorbike retail company with more than 30,000 dealerships and 2000 inspection centers globally. The company stores millions of records in Databricks hosted on Azure. You want to use Data Integration to perform some transformations on the data before you write back to Databricks.

Use a Databricks connection in the mapping to read from the Databricks source and write the processed data to the Databricks target. Configure full SQL ELT optimization in the mapping to enhance the performance.

Read from Amazon S3 and write to Databricks

You can configure SQL ELT optimization for a mapping that uses an Amazon S3 V2 connection in the Source transformation to read from Amazon S3 and a Databricks connection in the Target transformation to write to Databricks.

Example

You work for a healthcare organization. Your organization offers a suite of services to manage electronic medical records, patient engagement, telephonic health services, and care coordination services. The organization uses infrastructure based on Amazon Web Services and stores its data on Amazon S3. The management plans to load data to a data warehouse to perform healthcare analytics and create data points to improve operational efficiency. To load data from an Amazon S3 based storage object to Databricks, you must use ETL and ELT with the required transformations that support the data warehouse model.

Use an Amazon S3 V2 connection to read data from a file object in an Amazon S3 source and a Databricks connection to write to a Databricks target. Configure full SQL ELT optimization in the mapping to optimize the performance.

Read from Microsoft Azure Data Lake Storage Gen2 and write to Databricks

You can configure SQL ELT optimization for a mapping that uses an Microsoft Azure Data Lake Storage Gen2 connection in the Source transformation to read from Microsoft Azure Data Lake Storage Gen2 and a Databricks connection in the Target transformation to write to Databricks.

Example

You want to load data from an Microsoft Azure Data Lake Storage Gen2 based storage object to Databricks for analytical purposes. You want to transform the data before it is made available to users. Use an Microsoft Azure Data Lake Storage Gen2 connection to read data from a Microsoft Azure Data Lake Storage Gen2 source and a Databricks connection to write to a Databricks target. Configure full SQL ELT optimization in the mapping task to optimize the performance of loading data to Databricks. SQL ELT optimization enhances the performance of the task and reduces the cost involved.

SQL ELT compatibility

You can configure the task to push transformations, functions, and operators to the database.

When you use SQL ELT optimization, the Secure Agent converts the expression in the transformation by determining equivalent operators and functions in the database. If there is no equivalent operator and function, the Secure Agent processes the transformation logic.

Functions with Databricks

When you use SQL ELT optimization, Data Integration converts the expression in the transformation by determining equivalent functions in the database. If there is no equivalent function, Data Integration processes the transformation logic.

The following table summarizes the availability of SQL ELT functions that you can push to the Databricks database by using full or source SQL ELT optimization:

Function	Function
ABS()	MOD()
ADD_TO_DATE()	POWER()
ASCII()	RAND()
AVG()	REG_EXTRACT()
CEIL()	REG_MATCH()
CHR()	REG_REPLACE()
CONCAT()	REPLACESTR()
COS()	REPLACECHR()
COSH()	REVERSE()
COUNT()	ROUND(DATE)

Function	Function
CRC32()	ROUND(NUMBER)
DATE_COMPARE()	RPAD()
DATE_DIFF()	RTRIM()
DECODE()	SET_DATE_PART()
EXP()	SHA256()
FLOOR()	SIN()
GET_DATE_PART()	SINH()
GREATEST()	SQRT()
IIF()	STDDEV()
IN()	SUBSTR()
INDEXOF()	SUM()
INITCAP()	SYSDATE()
INSTR()	SYSTIMESTAMP()
IS_DATE()	TAN()
IS_NULL()	TANH()
IS_NUMBER()	TO_BIGINT
IS_SPACES()	TO_CHAR(DATE)
LAST()	TO_CHAR(NUMBER)
LAST_DAY()	TO_DATE()
LENGTH()	TO_DECIMAL()
LN()	TO_FLOAT()
LOWER()	TO_INTEGER()
LPAD()	TRUNC(DATE)
LTRIM()	TRUNC(NUMBER)
MAKE_DATE_TIME()	UPPER()
MAX()	VARIANCE()

Function	Function
MD5()	-
MIN()	-

Operators with Databricks

When you use SQL ELT optimization, the Secure Agent converts the expression in the transformation by determining equivalent operators in the database. If there is no equivalent operator, the Secure Agent processes the transformation logic.

The following table lists the SQL ELT operators that you can push to Databricks:

Operator	Operator
+	=
-	>=
*	<=
/	!=
%	AND
||	OR
>	NOT
<	

Variables with Databricks

You can use full SQL ELT optimization to push the SESSSTARTTIME and SYSDATE variable to the Databricks database.

SYSDATE is stored as a transformation date/time datatype variable. To return a static date and time, use the SESSSTARTTIME variable.

Configuring SQL ELT optimization

To optimize a mapping, add the mapping to a task, and then configure SQL ELT optimization in the mapping task.

1. Create a mapping task.
2. In the **SQL ELT optimization** section on the **Runtime Options** tab, set the SQL ELT optimization value to **Full** or **To Source**.

3. If full SQL ELT optimization is not available, select how Data Integration handles SQL ELT optimization in the **Fallback Option** menu:
   - Partial PDO. Default. Data Integration pushes as much transformation logic as possible to the source and target database. The task processes any transformation logic that it can't push to a database. You can use Partial PDO only when you read from and write to Databrick.
   - Non PDO. The task runs without SQL ELT optimization.
   - Fail Task. Data Integration fails the task.

   Note: The fallback options are not applicable to mappings in advanced mode.

When you run the mapping task, the transformation logic is pushed to the Databricks database.

Transformations with Databricks

When you configure SQL ELT optimization, the Secure Agent tries to push the configured transformation to Databricks.

You can use full or source SQL ELT optimization to push the following transformations to Databricks:

- Aggregator
- Expression
- Filter
- Joiner
- Lookup
- Sorter
- Union
- Router. Doesn't apply to source SQL ELT optimization.
- Rank
- SQL

Note: Expression, Filter, Lookup , Rank, and SQL transformations don't apply to mappings in advanced mode.

Aggregator transformation

You can configure full SQL ELT optimization to push an Aggregator transformation to process in Databricks.

Aggregate calculations

You can perform the following aggregate calculations:

- AVG
- COUNT
- FIRST
- LAST
- MAX
- MIN
- SUM
- STDDEV
- VARIANCE

Incoming ports

When you configure an Aggregator transformation and the incoming port is not used in an aggregate function or in a group by field, the output is not deterministic as the ANY_VALUE() function returns any value from the port.

You can pass only single arguments to the LAST, STDDEV, and VARIANCE functions.

Lookup transformation

You can configure full SQL ELT optimization to push a Lookup transformation to process in Databricks. This applies to both connected and unconnected lookups.

You can add the following lookups:

- Cached
- Uncached
- Unconnected with cached

When you configure a connected lookup, select the **Multiple Matches** property value as **Return all rows** in the lookup properties for SQL ELT optimization to work.

You can nest the unconnected lookup function with other expression functions.

When you configure an unconnected Lookup transformation, consider the following rules:

- You must select the **Multiple Matches** property value as **Report error** in the unconnected lookup properties for SQL ELT optimization to work.
- You can only configure an Expression transformation for an output received from an unconnected lookup.

SQL Transformation

You can use an SQL transformation to push supported scalar functions to Databricks.

When you configure SQL ELT optimization for a mapping, you can use scalar functions in a SQL transformation and run queries with the Databricks target endpoint.

You can use a simple SELECT statement without 'FROM' and 'WHERE' arguments. The SQL transformation only supports functions with simple SELECT statement.

The following snippet demonstrates the syntax of a simple SELECT SQL query:

```
SELECT <function_name1>(~Arg~), <function_name2> (~Arg~)...
```

For example, `SELECT SQRT(~AGE~)`

For more information about the supported functions, see the Databricks documentation.

Rules and guidelines for SQL transformation

Consider the following rules and guidelines when you use SQL transformation:

- You can configure only an SQL query in the SQL transformation. You cannot enable a stored procedure when you push down to Databricks.
- When you enable full SQL ELT optimization, ensure that you use the same connection type for the Source transformation and SQL transformation.
- When you specify a SELECT query, you must also specify the column name and number of columns based on the functions. For example, when you specify the query `select square(~AGE~),`

`sqrt(~SNAME~)`, you must specify two output columns for `AGE` and `SNAME` functions each, otherwise the mapping fails.

- If any SQL error occurs, the error is added to the `SQLError` field by default. However, when you run a mapping enabled with SQL ELT optimization, the `SQLError` field remains as Null.
- The `NumRowsAffected` field records the number of rows affected while computing the output buffer. However, for SQL transformation, the `NumRowsAffected` is 0, as the query runs for all the records at the same time.
- You cannot include special characters in the query, as SQL transformation does not support special characters in the arguments.
- You can use an SQL transformation when the SELECT statement is present only in the query property. You cannot configure an SQL transformation with a parameterized query, as dynamic parameter support is limited, and the query fails with a DTM error.

Features

You can configure SQL ELT optimization for a mapping that reads from the following sources and writes to a Databricks target:

- Databricks
- Amazon S3
- Microsoft Azure Data Lake Storage Gen2

When you configure a mapping, some parameters are not supported for a mapping enabled for SQL ELT optimization. You can refer to the list of parameters that each source supports.

Databricks sources, targets, and lookups

You must configure a Databricks connection with simple or hybrid mode when you enable SQL ELT optimization in a mapping task.

Source properties

When you configure SQL ELT optimization, the mappings support the following advance properties for a Databricks source:

- Source Object Type
  - Single
  - Multiple
  - Query
  - Parameter

  Note: When you use the query source type to read from Databricks, you can choose to retain the field metadata and save the mapping. Even if you edit the query and run the mapping, the field metadata specified at design time is retained.
- Query Options
  - Filter. You can use both simple and advanced filter conditions.
- Database Name
- Table Name
- SQL Override

Note: Contains, Ends With, and Starts With filter operators are not applicable when you use source filter to filter records.

Target properties

When you configure SQL ELT optimization, the mappings support the following properties for an Databricks target:

- Target Object Type
  - Single
  - Parameter
  - Create New at Runtime
- Operation
  - Insert
  - Update
  - Upsert
  - Delete
  - Data driven
- Create Target
- Target Database Name
- Target Table Name
- Update Mode
- Write Disposition for Insert operation.

Note: You cannot run pre-SQL or post-SQL commands in the source and target when you configure mappings for full SQL ELT optimization.

Lookup properties

When you configure SQL ELT optimization, the mappings support the following advance properties for a Databricks lookup:

- Source Object Type
  - Single
  - Query
  - Parameter
  - Multiple Matches for cached lookup

    Note: Un-cached unconnected lookup is not supported for SQL ELT optimization. For cached lookup, only **Return all rows** is supported.
- Database Name
- Table Name
- SQL Override

Note: If you configure advanced properties that are not supported, the Secure Agent either ignores the properties or logs a SQL ELT optimization validation error in the session logs file. If the Secure Agent logs an error in the session log, the mappings run in the Informatica runtime environment without full SQL ELT optimization.

Supported features for Amazon S3 V2 source

When you configure SQL ELT optimization, the mappings support the following properties for an Amazon S3 V2 source:

- Source connection parameter
- Source Type - Single, query
- Parameter
- Format - Avro, ORC, Parquet, JSON, and CSV
- Source Type - File and directory. XML source type is not applicable.
- Folder Path
- File Name

When you configure SQL ELT optimization, the mapping supports the following transformations:

- Filter
- Expression
- Aggregator
- Sorter
- Router
- Joiner
- Lookup
- Union
- Rank

For information on how to configure the supported properties, see the Amazon S3 V2 Connector documentation.

Supported features for Microsoft Azure Data Lake Storage Gen2 source

When you configure SQL ELT optimization, the Microsoft Azure Data Lake Storage Gen2 connection supports the following properties:

- Account Name
- Client ID
- Client Secret
- Tenant ID
- File System Name
- Directory Path
- Adls Gen2 End-point
- Server-side Encryption

When you configure SQL ELT optimization, the mappings support the following properties for a Microsoft Azure Data Lake Storage Gen2 source:

- Source connection, connection parameter
- Source Type - Single, parameter
- Format - Avro, Parquet, JSON, ORC, and CSV.

- Intelligent Structure Model
- Formatting Options
- Filesystem Name Override
- Source Type - File, Directory
- Directory Override - Absolute path; Relative path
- File Name Override - Source object
- Allow Wildcard Characters

When you configure SQL ELT optimization, the mapping supports the following transformations:

- Filter
- Expression
- Aggregator
- Sorter
- Router
- Joiner
- Lookup
- Union
- Rank

For information on how to configure the supported properties, see the Microsoft Azure Data Lake Storage Gen2 Connector documentation.

Configuring a custom query for the Databricks source object

You can push down a custom query to Databricks.

Before you run a task that contains a custom query as the source object, you must set the **Create Temporary View** session property in the mapping task properties.

Note: If you do not set the **Create Temporary View** property, the mapping runs without SQL ELT optimization.

Perform the following task to set the property:

1. In the mapping task, navigate to the **SQL ELT Optimization** section on the **Runtime Options** tab.
2. Select **Create Temporary View**.
3. Click **Finish**.

SQL ELT optimization for multiple targets

You can add multiple Databricks targets in a mapping.

When you configure a mapping to write to multiple Databricks targets, you can further optimize the write operation when you configure full SQL ELT optimization.

To optimize, you can configure an insert, update, upsert, or delete operation for multiple targets individually.

Note: You cannot configure an insert operation for unconnected target columns for mappings in advanced mode.

You can select the same Databricks target table in multiple Target transformations and perform different operations for each of the Target transformations to run independent of each other.

If your mapping contains multiple pipelines, you can define the flow run order to load the targets from the pipelines in a particular order.

Single commit for SQL ELT optimization

When you enable full SQL ELT optimization for a mapping to write to multiple Databricks targets, you can configure the mapping to commit the configured operations for all the targets within a connection group together.

You can use single commit to combine the metadata from all the targets and send the metadata for processing in a single execution call. When you use single commit, the Secure Agent separates the targets into connection groups based on equivalent connection attributes and commits the operations together for each connection group. This optimizes the performance of the write operation.

When you run a mapping with multiple targets, the Databricks connections used for these multiple target transformations that have the same connection attribute values are grouped together to form connection groups. As all the targets in a connection group have the same connection attributes, only a single connection is established for each connection group which represents that particular connection group. The transactions on each connection group runs on a single Databricks cluster.

If the Secure Agent fails to write to any of the targets, the task execution stops and the completed transactions for the targets that belong to the same connection group are not rolled back.

To enable single commit to write to multiple targets, set the **EnableSingleCommit=Yes** custom property in the **Advanced Session Properties** section on the **Runtime Options** tab of the mapping task.

When you run a mapping with single commit enabled, you can view the row statistics details in the session logs.

Single commit is applicable only when you run a mapping on Databricks cluster.

Rules and guidelines for SQL ELT optimization

Use the following rules and guidelines when you enable a mapping for SQL ELT optimization to a Databricks database:

General rules and guidelines

Use the following rules and guidelines when you configure SQL ELT optimization in a mapping:

- When you use the IN() function with OR, AND, or NOT operator in an IIF() function, the IIF() function returns incorrect results.
- When you configure mapping to read from an Amazon S3 or a Microsoft Azure Data Lake Storage Gen2 source, you cannot use Databricks connection in the Lookup transformation in the mapping.

Mapping with Databricks source and target

Use the following rules and guidelines when you configure SQL ELT optimization in a mapping that reads from and writes to Databricks:

- When you provide the database name in the connection, the following rules and guidelines apply:
  - For full SQL ELT optimization, temporary staging tables or views are created in the database name provided in the target connection attribute `database`. If the attribute is empty the database name is picked from JDBC URL of the target connection.

- For Source SQL ELT optimization, temporary staging tables or views are created in the database name provided in the source connection attribute `database`. If the attribute is empty the database name is picked from JDBC URL of source connection.
- The database name provided in the connection should have read and write permission. If the `database name` field is empty, then the database name provided in the JDBC URL will take precedence and should have read and write permission.

Note: The guidelines are only applicable to custom query support, SQL override, and SQL ELT optimization in Databricks using flat file formats.

- You cannot use the Data Driven target operation with the Update Mode advanced target property.
- LAST function is a non-deterministic function. This function returns different results each time it is called, even when you provide the same input values.
- When you configure a Filter transformation or specify a filter condition, do not specify special characters.
- When you connect to Databricks clusters to process the mapping and define a custom query with multiple tables in the SELECT statement, the mapping displays incorrect data for fields that have the same name. This doesn't apply to Databricks runtime version 9.1 LTA or later.
- When you configure a mapping enabled for full SQL ELT optimization to read from multiple sources and you override the database name and table name from the advanced properties, the mapping fails.
- To configure a Filter transformation or specify a filter condition on columns of date or timestamp in a Databricks table, you must pass the data through the TO_DATE() function as an expression in the filter condition.
- When you specify custom query as a source object, ensure that the SQL query does not contain any partitioning hints such as COALESCE, REPARTITION, or REPARTITION_BY_RANGE.
- When you configure a mapping enabled for full SQL ELT optimization on the Databricks SQL engine, you cannot configure single commit to write to multiple targets.
- When you configure a mapping enabled for full SQL ELT optimization on the Databricks SQL engine and push the data to the Databricks target, ensure that you map all the fields in target. Else, the mapping fails.
- When you create a new target at runtime, you must not specify a database name and table name in the **Target Database Name** and **Target Table Name** in the target advanced properties.
- When you read data from a column of Date data type and write data into a column of Date data type, the SQL ELT query pushes the column of Date data type and casts the column to Timestamp data type.
- You cannot completely parameterize a multi-line custom query using a parameter file. If you specify a multi-line custom query in a parameter file, the mapping considers only the first line of the multi-line query.
- When you push the CRC32() funtion to Databricks, the data type of the return value is either Bigint or String.
- When you push the DATE_DIFF() function to Databricks, the function returns the integral part of the value and not the fractional part.
- When you push the GREATEST() function to Databricks and configure input value arguments of String data type, you must not specify the caseFlag argument.
- To push the TO_CHAR(DATE) function to Databricks, use the following string and format arguments:
  - YYYY
  - YY
  - MM
  - MON

- MONTH
- DD
- DDD
- DY
- DAY
- HH12
- HH24
- MI
- Q
- SS
- SS.MS
- SS.US
- SS.NS

- To push the TO_DATE(string, format) function to Databricks, you must use the following format arguments:
  - YYYY
  - YY
  - MM
  - MON
  - MONTH
  - DD
  - DDD
  - HH12
  - HH24
  - MI
  - SS
  - SS.MS
  - SS.US
  - SS.NS
- When you enable full SQL ELT optimization in a mapping and use the IFF() condition in an Expression transformation, the mapping fails for the following functions:
  - IS_SPACES
  - IS_NUMBER
  - IS_DATE
- A mapping enabled with full SQL ELT optimization and contains an SQL transformation fails when the column names in the SQL override query don't match with the column names in the custom query.

Mapping with Amazon S3 source and Databricks target

Use the following rules and guidelines when you configure SQL ELT optimization in a mapping that reads from an Amazon S3 source and writes to a Databricks target:

- When you select the source type as directory in the advanced source properties, ensure that all the files in the directory contain the same schema.
- When you select query as the source type in lookup, you cannot override the database name and table name in the advanced source properties.
- When you include a source transformation in a mapping enabled with SQL ELT optimization, exclude the FileName field from the source. The FileName field is not applicable.
- When you parameterize a lookup object in a mapping enabled with SQL ELT optimization, the mapping fails as you cannot exclude the filename port at runtime.
- When you parameterize the source object in a mapping task, ensure that you pass the source object parameter value with the fully qualified path in the parameter file.
- You cannot use wildcard characters for the source file name and directory name in the source transformation.
- You cannot use wildcard characters for the folder path or file name in the advanced source properties.
- When you read from a partition folder that has a transaction log file, select the source type as Directory in the advanced source properties.
- You cannot configure dynamic lookup cache.
- When you use a Joiner transformation in a mapping enabled with SQL ELT optimization and create a new target at runtime, ensure that the fields do not have a not null constraint.
- Ensure that the field names in Parquet, ORC, AVRO, or JSON files do not contain Unicode characters.

Mapping with Azure Data Lake Storage Gen2 source and Databricks target

Use the following rules and guidelines when you configure SQL ELT optimization in a mapping that reads from a Azure Data Lake Storage Gen2 source and writes to a Databricks target:

- Mappings fail if the lookup object contains unsupported data types.
- When you select the source type as directory in the advanced source property, ensure that all the files in the directory contain the same schema.
- When you select query as the source type in lookup, you cannot override the database name and table name in the advanced source properties.
- When you include a source transformation in a mapping enabled with SQL ELT optimization, exclude the FileName field from the source. The FileName field is not applicable.
- When you parameterize a lookup object in a mapping enabled with SQL ELT optimization, the mapping fails as you cannot exclude the filename port at runtime.
- When you parameterize the source object in a mapping task, ensure that you pass the source object parameter value with the fully qualified path in the parameter file.
- You cannot use wildcard characters for the source file name and directory name in the source transformation.
- When you read from a partition folder that has a transaction log file, select the source type as Directory in the advanced source properties.
- You cannot configure dynamic lookup cache.
- When you use a Joiner transformation in a mapping enabled with SQL ELT optimization and create a new target at runtime, ensure that the fields do not have a not null constraint.

- Ensure that the field names in Parquet, ORC, AVRO, or JSON files do not contain Unicode characters.

Cross workspace mappings

When you set up a mapping enabled with full SQL ELT optimization to access data from a Databricks workspace, and the associated metastore resides in a separate workspace, the mapping runs without SQL ELT optimization.

Troubleshooting SQL ELT optimization

Mapping fails when configured to read date or timestamp information and write to default date/time format

When you configure a mapping to read date or timestamp information from a string column and process the data with the default date/time format to write to Databricks target, the mapping fails with the following error:

```
[ERROR] The Secure Agent failed to run the full pushdown query due to the following error: [Invalid timestamp: '12/31/1972 00:00:00.000001']
```

To resolve this issue, set the JVM option `-DHonorInfaDateFormat=true` for the Secure Agent.

Perform the following steps to configure the JVM option in Administrator:

1. Select **Administrator** > **Runtime Environments**.
2. On the Runtime Environments page, select the Secure Agent machine that runs the mapping.
3. Click **Edit**.
4. In the System Configuration Details section, select **Data Integration Server** as the Service and **DTM** as the Type.
5. Edit the JVMOption system property and set the value to `-DHonorInfaDateFormat=true`.
6. Click **Save**.

IS_DATE(), IS_SPACES(), and IS_NUMBER() functions return 0 or 1 instead of True or False.

When you use IS_DATE(), IS_SPACES(), and IS_NUMBER() functions, the functions return 0 or 1 instead of True or False.

To resolve this issue, set the JVM option `-DDeltaSQLELTBooleanReturnAsString=true` for the Secure Agent.

Perform the following steps to configure the JVM option in Administrator:

1. Select **Administrator** > **Runtime Environments**.
2. On the Runtime Environments page, select the Secure Agent machine that runs the mapping.
3. Click **Edit**.
4. In the System Configuration Details section, select **Data Integration Server** as the Service and **DTM** as the Type.
5. Edit the JVMOption system property and set the value to `-DDeltaSQLELTBooleanReturnAsString=true`.
6. Click **Save**.

CHAPTER 6

Data type reference

Databricks native data types

Databricks data types appear in the Fields tab for Source and Target transformations when you choose to edit metadata for the fields.

Transformation data types

Set of data types that appear in the remaining transformations. They are internal data types based on ANSI SQL-92 generic data types, which Data Integration uses to move data across platforms. Transformation data types appear in all remaining transformations in Data Integration tasks.

When the Data Integration application reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When the Data Integration application writes to a target, it converts the transformation data types to the comparable native data types.

Databricks and transformation data types

The following table compares the Databricks native data type to the transformation data type:

Databricks Data Type	Transformation Data Type	Range and Description
Array[1]	Array	Unlimited number of characters.
Binary	Binary	1 to 104,857,600 bytes.
Bigint	Bigint	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807. 8-byte signed integer.
Boolean	Integer	1 or 0.
Date	Date/Time	Date and time values.
Decimal	Decimal	Exact numeric of selectable precision For mappings, the maximum precision is 28 and the scale is 27. For mappings in advanced mode, the maximum precision is 38 and the scale is 37.

Databricks Data Type	Transformation Data Type	Range and Description
Double	Double	Precision 15.
Float	Double	Precision 7.
Int	Integer	-2,147,483,648 to +2,147,483,647.
Map[1]	Map	Unlimited number of characters.
Smallint	Integer	-32,768 to +32,767.
String	String	1 to 104,857,600 characters.
Struct[1]	Struct	Unlimited number of characters.
Tinyint	Integer	-128 to 127
Timestamp	Date/Time	January 1,0001 00:00:00 to December 31,9999 23:59:59.997443. Timestamp values only preserve results up to microsecond precision of six digits. The precision beyond six digits is discarded.
[1]Applies only to mappings in advanced mode. Doesn't apply to Databricks cluster and SQL ELT optimization.		

Rules and guidelines for data types

Databricks has the following restriction for decimal data type:

The behavior of the decimal data type differs in mappings and mappings in advanced mode. Mappings support a maximum decimal precision of 28, while mappings in advanced mode support a maximum decimal precision of 38.

In mappings, if the decimal data type exceeds the precision of 28 in the source, the numeric value for the decimal is rounded off after the 18th precision and the remaining digits are replaced with zeroes in the target.

For example, the value 12345678901234567890123456789.012345678 from the source is rounded off to 12345678901234569000000000000 in the target. However, in mappings in advanced mode, the decimal data from the source remains the same in the target.

To resolve the issue in mappings, specify a precision that is less than or equal to 28 for the Decimal data type in the source table.

INDEX

C

Cloud Application Integration community
  URL 6
Cloud Developer community
  URL 6
Create target
  rules and guidelines 37
create target at runtime 36
custom query 31

D

Data Integration community
  URL 6
data types 79
Databricks
  SQL ELT optimization 67
  SQL ELT optimization overview 63
Databricks connections
  overview 10
Databricks connector
  datetime format 47
  rules and guidelines 47
Databricks Connector
  assets 8
  overview 8
Databricks connector rules and guidelines 46
Databricks Delta 28

F

field delimiter 28

I

Informatica Global Customer Support
  contact information 7
Informatica Intelligent Cloud Services
  web site 6

M

maintenance outages 7
mappings
  overview 25
  source properties 28
mappings *(continued)*
  target properties 33
mappings in advanced mode
  example 45

N

native data type 79

P

properties
  in mappings 28, 33

S

source properties 28
SQL ELT optimization
  functions 65, 67
  preview 55
  transformations 65, 67, 68
SQL ELT Optimization
  rules and guidelines 74
SQL ELT optimization preview 55
status
  Informatica Intelligent Cloud Services 7
system status 7

T

tracing level 28
transformation data type 79
transformations
  SQL ELT optimization 68
trust site
  description 7

U

upgrade notifications 7

W

web site 6