



TRIFACTA

Connection Guide

Version: 9.2
Doc Build Date: 07/29/2022

Copyright © Trifacta Inc. 2022 - All Rights Reserved. CONFIDENTIAL

These materials (the “Documentation”) are the confidential and proprietary information of Trifacta Inc. and may not be reproduced, modified, or distributed without the prior written permission of Trifacta Inc.

EXCEPT AS OTHERWISE PROVIDED IN AN EXPRESS WRITTEN AGREEMENT, TRIFACTA INC. PROVIDES THIS DOCUMENTATION AS-IS AND WITHOUT WARRANTY AND TRIFACTA INC. DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES TO THE EXTENT PERMITTED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE AND UNDER NO CIRCUMSTANCES WILL TRIFACTA INC. BE LIABLE FOR ANY AMOUNT GREATER THAN ONE HUNDRED DOLLARS (\$100) BASED ON ANY USE OF THE DOCUMENTATION.

For third-party license information, please select **About Trifacta** from the Help menu.

1. Connect	4
1.1 Connection Types	5
1.1.1 Oracle Database Connections	9
1.1.2 PostgreSQL Connections	13
1.1.3 MySQL Connections	17
1.1.4 Microsoft SQL Server Connections	21
1.1.5 Google Sheets Connections	25
1.1.6 External S3 Connections	27
1.1.7 Amazon Redshift Connections	34
1.1.8 AWS Glue Connections	41
1.1.9 Snowflake Connections	46
1.1.10 Azure SQL Database Connections	52
1.1.11 Microsoft SQL Data Warehouse Connections	54
1.1.12 Databricks Tables Connections	58
1.1.13 SFTP Connections	65
1.1.14 REST API Connections	70
1.1.15 Enable Teradata Access	78
1.1.16 Teradata Connections	80
1.1.17 MongoDB Connections	84
1.1.18 Tableau Server Connections	89
1.1.19 Salesforce Connections	92
1.1.20 SharePoint Connections	97
1.1.21 Google Analytics Connections	102
1.1.22 DB2 Connections	105
1.2 Configure Connectivity	108
1.2.1 Relational Access	111
1.2.2 Enable Custom SQL Query	114
1.2.3 Configure JDBC Ingestion	116
1.2.4 Configure Data Source Caching	120
1.2.5 Configure Security for Relational Connections	123
1.2.6 Enable SSO for Relational Connections	127
1.2.7 Enable OAuth 2.0 Authentication	131
1.2.7.1 Create OAuth2 Client	134
1.2.7.2 OAuth 2.0 for Google Analytics	136
1.2.7.3 OAuth 2.0 for Google Sheets	139
1.2.7.4 OAuth 2.0 for Salesforce	142
1.2.7.5 OAuth 2.0 for Snowflake	145
1.2.8 Configure Connectivity for Amazon RDS	149
1.2.9 Configure Type Inference	150
1.2.10 Troubleshooting Relational Connections	153
1.2.11 Supported Connection Credential Types	155

Connect

Contents:

- *Not Covered*
 - *Hadoop*
 - *AWS*
-

This section covers how to configure JDBC integration for Trifacta® and connect your working platform instance to a wide variety of JDBC-based connections.

Many of these connections can be created from the Trifacta application directly. In some cases, additional configuration is required outside of the application.

Not Covered

This guide does not cover connection types that are deeply tied to a specific infrastructure. The following connection types are described in the appropriate Configuration Guide.

Hadoop

- *Hive Connections*

AWS

- *S3 Access*
- *Amazon Redshift Connections*

Connection Types

Contents:

- *Configure*
 - *Disable Creating Connections for Non-Admins*
 - *Create Connection*
 - *Connection Categories*
 - *Applications*
 - *CRM ERP*
 - *Databases*
 - *File-API*
 - *Marketing*
 - *Connections to Base Storage Layer*
 - *Apache Hadoop HDFS - Cloudera*
 - *Amazon AWS*
 - *Microsoft Azure*
 - *Amazon S3 (layer)*
 - *Default Connections*
 - *Upload*
 - *Other Connections*
 - *Hive*
 - *Custom connections*
-

You can create the following types of connections from Trifacta®. You can use the given links to create connection.

Notes:

- HDFS and Hive connections can be configured as part of platform configuration.
- Database connections should be configured after you have completed the platform configuration and have validated that it is working for locally uploaded files.

NOTE: Before creating connections to Hive or relational datastores, you must create and deploy an encryption key file. See *Create Encryption Key File*.

Configure

Disable Creating Connections for Non-Admins

By default, all users are permitted to create connections. As needed, you can disable the ability to create connections for non-admin users.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Search for the following parameter, and set it to `false`:

```
"webapp.connectivity.nonAdminManagementEnabled": true,
```

3. Save your changes and restart the platform.

Create Connection

1. Click **Connections** in the left nav bar.
2. In the Connections page, click **Create**.

Actions:

Search: Enter a search string in the search bar to locate connections of interest.

Tip: Search scans the selected category. For broadest search results, select the All types tab.

I'm interested: Click I'm interested to upvote adding the connection type to the Trifacta application. Help chart the future direction of connectivity!

For more information, see *Create Connection Window*.

Connection Categories

In the Create Connection window, connections are organized according to the following categories.

Applications

Connect to data storage for your web-based applications.

CRM ERP

These connections enable you to access data in your CRM or ERP systems.

Databases

These connections pertain to relational database sources.

NOTE: Unless otherwise noted, authentication to a relational connection requires basic authentication (username/password) credentials.

Enable: For more information, see *Relational Access*.

File-API

You can create connections to the listed file-based datastores or to other API-based storage.

Marketing

You can create connections to the datastores for popular marketing solutions.

Connections to Base Storage Layer

Connections of these type are automatically created for you.

Apache Hadoop HDFS - Cloudera

Enable: *Configure for Hadoop.*

Create New Connection: n/a

Amazon AWS

Running environment(s): Trifacta Photon and Spark

Base storage layer: S3

Microsoft Azure

Running environment(s): Trifacta Photon and Spark

Base storage layer: ADLS Gen1, ADLS Gen2, or WASB

For more information, see *Running Environment Options*.

For more information, see *Set Base Storage Layer*.

Amazon S3 (layer)

This connection type refers to using S3 as the base storage layer.

Supported Versions: n/a

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Not supported	Supported	Not supported

Enable: S3 Access

Create New Connection: n/a

Default Connections

These connections are automatically enabled and configured with the product.

Upload

Enable: Automatically enabled.

Create New Connection: n/a

Other Connections

Hive

Enable: *Configure for Hive*

NOTE: Additional configuration is required.

Create New Connection:

NOTE: A single public Hive connection is supported.

For more information, see *Hive Connections*.

Custom connections

For more information on other connectivity options, please contact *Alteryx Support*.

Oracle Database Connections

Contents:

- *Prerequisites*
 - *SSL*
 - *Configure*
 - *Connection URL*
 - *Driver Information*
 - *Create via API*
 - *Troubleshooting*
 - *"Could not create dataset - Lexical error"*
 - *Use*
 - *SQL Syntax*
 - *Data Conversion*
-

You can create connections to one or more Oracle Database from Trifacta®.

Tip: You can create connections to databases of this type that are managed by your enterprise or are hosted in cloud infrastructure. The required configuration is the same. The cloud-based version is labeled **on Amazon RDS**. In the Create Connection dialog, you can search for that term.

Supported Versions: 12.1.0.2

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Supported	Supported	Supported

Prerequisites

NOTE: Dots (.) in the names of Oracle tables or table columns are not supported.

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

SSL

If you are connecting to the Oracle Database using SSL, additional configuration is required in Oracle Database.

Trifacta supports the use of the following SSL ciphers to communicate with Oracle Database:

For more information, please see the documentation that is provided with your Oracle distribution.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the Oracle Database card
- You can also create connections through the Connections page. See *Connections Page*.

For additional details on creating an Oracle Database connection, see *Relational Access*.

Modify the following properties as needed:

Property	Description
Host	Enter your hostname. Example: <div>testsql.database.windows.net</div>
Port	Set this value to 1521.
Connect String options	Please insert any connection options as a string here. See below.
Enable SSL	Select the option if the connection should use SSL. <div>NOTE: Additional configuration may be required in the database server. For more information, please consult the documentation that was provided with the distribution.</div> <div>NOTE: Additional configuration is required. See <i>Configure Data Service</i>.</div>
Service Name	Enter the name of the Oracle service.
User Name	(basic credential type only) Username to use to connect to the database.
Password	(basic credential type only) Password associated with the above username.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Connection Name	Display name of the connection

Connection Description	Description of the connection, which appears in the application.
------------------------	--

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
<host>:<port>/<service_name>
```

Connect string options

The connect string options are optional. If non-standard connections are required, Oracle Database supports using tsnames format.

When the connect string options field is used:

- The connect string options parameters are prepended with `jdbc:oracle:thin:@`.
- The following fields are ignored from the form. These values must be specified as part of the tsnames :
 - Host
 - Port
 - Service
 - SSL

After you specify the connect string options, the generated connection URL is automatically prepended with the following protocol information. Do not add this to the connection URL or connect string options:

```
jdbc:oracle:thin:@
```

Examples are below.

Use SID:

If you are using a service identifier, instead of a service name, please specify your connection string options as follows:

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=oracle.rds.example.com)(PORT=1521))(CONNECT_DATA=(SID=orcl)))
```

Use TCPS:

If TCPS protocol is required, you can specify your connection string options as follows:

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=oracle.rds.example.com)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=orcl)))
```

For more information, please see the documentation for the Oracle Database driver.

Driver Information

This connection uses the following driver:

- **Driver name:** `oracle.jdbc.driver.OracleDriver`
- **Driver version:** `com.oracle.database.jdbc:ojdbc8:19.9.0.0`
- **Driver documentation:** <https://docs.oracle.com/en/database/oracle/oracle-database/19/index.html>

Create via API

This connection can also be created using the API.

- Type: jdbc
- Vendor: oracle

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Troubleshooting

For more information on common error messages, see https://docs.oracle.com/cd/E11882_01/java.112/e16548/apxerrmsg.htm#JJDBC28962.

"Could not create dataset - Lexical error"

When you attempt to create a dataset with SQL from an Oracle database, you may receive an ORA error similar to the above. These queries may work in other database tools.

Solution:

The solution is to apply aliasing impacted columns in your SQL query. For more information, see *Supported SQL Syntax*.

Use

SQL Syntax

The following syntax requirements apply to this connection.

Object delimiter: double-quote

Example syntax:

Double quotes required around database and table names and not required around column names.

```
SELECT "column1","column2" FROM "databaseName"."tableName";
```

For more information on SQL in general, see *Supported SQL Syntax*.

Data Conversion

For more information on how values are converted during input and output with this database, see *Oracle Data Type Conversions*.

PostgreSQL Connections

Contents:

- *Prerequisites*
 - *Configure*
 - *Connection URL*
 - *Driver Information*
 - *Create via API*
 - *Troubleshooting*
 - *Use*
 - *SQL Syntax*
 - *Data Conversion*
-

You can create connections to a PostgreSQL database from Trifacta®. For more information on PostgreSQL, see <https://www.postgresql.org/>.

Tip: You can create connections to databases of this type that are managed by your enterprise or are hosted in cloud infrastructure. The required configuration is the same. The cloud-based version is labeled **on Amazon RDS**. In the Create Connection dialog, you can search for that term.

Supported Versions: 9.3.10

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Prerequisites

If the Trifacta databases are hosted on a PostgreSQL server, do not create a connection to this database.

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the PostgreSQL card.
- You can also create connections through the Connections page. See *Connections Page*.

For additional details on creating a PostgreSQL connection, see *Relational Access*.

Modify the following properties as needed:

Property	Description
Host	Enter your fully qualified hostname. Example: <div>my.postgres.server</div>
Port	Set this value to 5432.
Connect String Options	Insert any additional connection parameters, if needed. See below.
Enable SSL	Select the checkbox to enable SSL connections to the database. <div>NOTE: Additional configuration may be required in the database server. For more information, please consult the documentation that was provided with the distribution.</div>
Database	Enter the name of the database on the server to which to connect.
User Name	Username to use to connect to the database.
Password	Password associated with the above username.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Default Column Data Type Inference	Set to <code>disabled</code> to prevent the platform from applying its own type inference to each column on import. The default value is <code>enabled</code> .
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:postgresql://<host>:<port>/<database><connect-string-options>
```

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
?<prop1>=<val1>&<prop2>=<val2>...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

delimiters:

- `?` : any set of connect string options must begin with a question mark.
- `&` : all additional property names must be prefixed with an ampersand (&).
- `=` : property names and values must be separated with an equal sign (=).

Driver Information

This connection uses the following driver:

- **Driver name:** `org.postgresql.Driver`
- **Driver version:** `org.postgresql:postgresql:42.1.1`
- **Driver documentation:** <https://jdbc.postgresql.org/documentation/head/index.html>

Create via API

This connection can also be created using the API.

- Type: `jdbc`
- Vendor: `postgres`

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Troubleshooting

Error message	Description
Class 08 Connection Exception	Connection failure: the web client or Trifacta node is unable to establish a connection.
Class 28 Invalid Authorization Specification	Typically, this error occurs when an invalid password has been submitted. <div>Tip: Use the Test Connection button to validate your credentials.</div>

For more information on error messages for this connection type, see <https://www.postgresql.org/docs/9.3/errcodes-appendix.html>.

NOTE: Please note the version number in the URL above.

Use

For more information, see *Database Browser*.

For more information on interacting with data, see *Using Databases*.

SQL Syntax

The following syntax requirements apply to this connection.

Object delimiter: double-quote

Example syntax:

Double quotes required around database, table names, and column names.

```
SELECT "column1", "column2" FROM "databaseName"."tableName";
```

For more information on SQL in general, see *Supported SQL Syntax*.

Data Conversion

For more information on how values are converted during input and output with this database, see *Postgres Data Type Conversions*.

MySQL Connections

Contents:

- *Prerequisites*
 - *Configure*
 - *Connection URL*
 - *Driver Information*
 - *Create via API*
 - *Troubleshooting*
 - *Use*
 - *Data Conversion*
-

You can create connections to one or more MySQL databases from Trifacta®. For more information on MySQL, see <https://www.mysql.com/>.

Tip: You can create connections to databases of this type that are managed by your enterprise or are hosted in cloud infrastructure. The required configuration is the same. The cloud-based version is labeled **on Amazon RDS**. In the Create Connection dialog, you can search for that term.

Supported Versions: 5.7 and 8.0 Community

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Prerequisites

If the Trifacta databases are hosted on a MySQL server, do not create a connection to this database.

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the MySQL card.
- You can also create connections through the Connections page.
- See *Connections Page*.

For additional details on creating a MySQL connection, see *Relational Access*.

Modify the following properties as needed:

Property	Description
Host	Enter your fully qualified hostname. Example: <div>mysql-server.example.net</div>
Port	Set this value to 3306.
Connect String Options	Insert any additional connection parameters, if needed. See below.
User Name	Username to use to connect to the database.
Password	Password associated with the above username.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Default Column Data Type Inference	Set to disabled to prevent the platform from applying its own type inference to each column on import. The default value is enabled.
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:mysql://<host>:<port>/<database><connect-string-options>
```

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
&<prop1>=<val1>&<prop2>=<val2>...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

delimiters:

- `&` : all additional property names must be prefixed with an ampersand (&).
- `=` : property names and values must be separated with an equal sign (=).

Default connect string options

The following connect string options are specified by default.

NOTE: These options should not be overridden or modified.

The following connect string option requires the driver to use cursor-based fetching to retrieve rows.

```
useCursorFetch=true;
```

Enable TLS (SSL)

You can insert the following connection string option to enable secure (TLS) connectivity with the MySQL server. Please note the TLS version numbers in the string listed below:

```
&enabledTLSProtocols=TLSv1,TLSv1.1,TLSv1.2
```

Set time zone

When data is imported from MySQL, the connector performs time zone adjustments to imported Datetime values.

Tip: By default, the MySQL driver adjusts the imported data to be represented in the time zone into which it is being read. If you are located in GMT -08:00, all data is rendered that your Trifacta user reads into the application through the MySQL connector is adjusted to this time zone.

These adjustments are performed in either of the following cases:

- The MySQL server is configured with a canonical time zone that is recognizable by Java (for example, Europe/Paris, Etc/GMT-5, UTC, etc.).
- The server's time zone is overridden by setting the Connector/J connection property `serverTimezone`.

In the latter case, you can specify the override time zone to apply as part of the connection string:

```
&serverTimezone=Europe/Paris
```

The following value sets the time zone of imported data to be UTC:

```
&serverTimezone=UTC
```

For more information on this behavior, see <https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-other-changes.html>.

Driver Information

This connection uses the following driver:

- **Driver name:** `com.mysql.cj.jdbc.Driver`
- **Driver version:** `8.0`
- **Driver documentation:** <https://dev.mysql.com/doc/connector-j/8.0/en/>

Create via API

This connection can also be created using the API.

- Type: `jdbc`
- Vendor: `mysql`

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Troubleshooting

Error message	Description
1042 - ER_BAD_HOST_ERROR	Unable to connect to host. Please verify the host and port values.

1045 - ER_ACCESS_DENIED_ERROR	<p>Credentials failed to connect. Please verify your credentials.</p> <div> Tip: Click the Test Connection button to verify that your credentials are working properly. </div>
Error: zero date value prohibited	<p>Set the following option in the connect string options:</p> <pre>zeroDateTimeBehavior=convertToNull</pre>
Prepared statement needs to be re-prepared.	<p>Database Cursor is not compatible with PREPARED statements in MySQL. The fix is to set the following in the Connect String Options:</p> <pre>useCursorFetch=false</pre>
SSLHandshakeException: No appropriate protocol (protocol is disabled or cipher suites are inappropriate)	<p>SSL ciphers need to be enabled. For more information, see "Enable TLS (SSL)" above.</p>

For more information on error messages for this connection type, see <https://dev.mysql.com/doc/refman/8.0/en/error-handling.html>.

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *MySQL Data Type Conversions*.

Microsoft SQL Server Connections

Contents:

- *Prerequisites*
 - *Configure*
 - *Connection URL*
 - *Authentication options*
 - *Driver Information*
 - *Create via API*
 - *Troubleshooting*
 - *Use*
 - *SQL Syntax*
 - *Data Conversion*
-

You can create connections to one or more Microsoft SQL Server databases from Trifacta®.

Tip: You can create connections to databases of this type that are managed by your enterprise or are hosted in cloud infrastructure. The required configuration is the same. The cloud-based version is labeled **on Amazon RDS**. In the Create Connection dialog, you can search for that term.

Supported Versions: 12.0.4

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Prerequisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.
- If you plan to create an SSO connection of this type, additional configuration may be required. See *Enable SSO for Relational Connections*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the **Microsoft SQL Server** card.
- You can also create connections through the Connections page. See *Connections Page*.

For additional details on creating a Microsoft SQL Server connection, see *Relational Access*.

Modify the following properties as needed:

Property	Description
----------	-------------

Host	Enter your hostname. Example: <code>testsql.database.windows.net</code>
Port	Set this value to 1433.
Connect String options	Insert any additional connection parameters, if needed. See below.
User Name	(basic credential type only) Username to use to connect to the database.
Password	(basic credential type only) Password associated with the above username.
Default Column Data Type Inference	Set to disabled to prevent the platform from applying its own type inference to each column on import. The default value is enabled.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:sqlserver://<host>:<port>;<prop1>=<val1>;<prop2>=<val2>
```

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
;<prop1>=<val1>;<prop2>=<val2>...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

delimiters:

- `;` : any set of connect string options must begin and end with a semi-colon.
- `=` : property names and values must be separated with an equal sign (=).

Example connect string options

The following connect string contains several options. Please insert as a single string (no line breaks):

```
;database=<database_name>;encrypt=true;trustServerCertificate=false;  
hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

Common connect string properties:

Property	Description
database	Set this value to <code><database_name></code> , the name of the database to which to connect.

encrypt	Set this value to true. Encrypted communication is required. NOTE: You must deploy an encryption key file on the Trifacta node. For more information, see <i>Create Encryption Key File</i> .
trustServerCertificate	When set to true, the Trifacta node does not validate the SQL Server TLS/SSL certificate. Default value is false.
hostNameInCertificate	Defines the host name in the server certificate. NOTE: Do not modify this value unless required.
loginTimeout	Number of seconds that the Trifacta node attempts to login to the database server. Default is 30.

Delimiters:

- ; : any set of connect string options must begin and end with a semi-colon.
- ; : all additional property names must be prefixed with a semi-colon.
- = : property names and values must be separated with an equal sign (=).

Authentication options

Kerberos:

You can use kerberos security for connecting to the database server. Additional configuration is required:

- For more information on enabling single-sign on for databases, see *Enable SSO for Relational Connections*.
- The SSO solution for databases leverages the platform's integration with cluster kerberos. For more information, *Configure for Kerberos Integration*.

Driver Information

This connection uses the following driver:

- **Driver name:** `com.microsoft.sqlserver.jdbc.SQLServerDriver`
- **Driver version:** `com.microsoft.sqlserver:mssql-jdbc:7.2.2.jre8`
- **Driver documentation:**
 - Overview:
<https://docs.microsoft.com/en-us/sql/connect/jdbc/overview-of-the-jdbc-driver?view=sql-server-ver15>
 - Connection URL:
<https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url?view=sql-server-ver15>
 - Connection properties:
<https://docs.microsoft.com/en-us/sql/connect/jdbc/setting-the-connection-properties?view=sql-server-ver15>

Create via API

This connection can also be created using the API.

- Type: `jdbc`
- Vendor: `sqlserver`

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Troubleshooting

Error message	Description
"The TCP/IP connection to the host <hostname>, port <port> has failed"	The host is not accessible.
"Login failed for user '<username>'."	Permission denied. Please verify your credentials. <div>Tip: Click the Test Connection button to check the connection credentials.</div>
"The certificate received from the remote server was issued by an untrusted certificate authority"	There was an issue with the trusted certificate on the SQL Server instance. To disable validation of the certificate, add the following to the connection string options: <div><code>;trustServerCertificate=true;</code></div>

Use

SQL Syntax

The following syntax requirements apply to this connection.

Object delimiter: none

Example syntax:

```
SELECT "column1","column2" FROM "databaseName"."tableName";
```

For more information on SQL in general, see *Supported SQL Syntax*.

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *SQL Server Data Type Conversions*.

Google Sheets Connections

Contents:

- *Limitations*
- *Enable*
- *Configure*
- *Use*
- *Testing*

Google Sheets provides a spreadsheet interface for cloud-based data stored in Google Drive. It allows multiple users to edit and format files in real-time. For more information, see <https://www.google.com/sheets/about/>.

- You can create multiple Google Sheets connections.
- During connection creation, you must provide access to Google Drive.

You can create connections to your Google Sheets instance from Trifacta®.

Limitations

NOTE: This connection type is not supported for integration with a Hadoop-based running environment.

- This is a read-only connection.
- Single Sign-On (SSO) is not supported.
- Custom domains are not supported.
- Since data must be converted to a native file format, this connection does not support previewing of your data in the source.
- If you have enabled Google Advanced Protection, this connection type does not work.

Enable

- General relational connectivity must be enabled. For more information, see *Relational Access*.
- This connection type utilizes OAuth 2.0 for authentication. For more information, see *Enable OAuth 2.0 Authentication*.
- An OAuth 2.0 web client must be available for use in the Trifacta application. For more information, see *OAuth 2.0 for Google Sheets*.

Configure

To create this connection, select the **Google Sheets** card. For more information, see *Connections Page*.

Modify the following properties as needed:

Property	Description
OAuth 2.0 Client	The OAuth 2.0 Client is displayed. <div>NOTE: When you create the connection in this window, you must click Authenticate, which authenticates to the app. This step is required.</div>
Connection Name	Display name of the connection.

Connection Description	Description of the connection, which appears in the application.
---------------------------	--

Use

You can import datasets from Google Sheets through the Import Data page. For more information, see *Import Google Sheets Data*.

Testing

Steps:

1. After you create your connection, import datasets from the connected Google Sheets. For more information, see *Import Data Page*.
2. Perform a few simple transformations to the data. Run the job. For more information, see *Transformer Page*.
3. Verify the results.

External S3 Connections

Contents:

- *Prerequisites*
 - *Permissions*
 - *Limitations*
 - *Create Connection*
 - *Create through application*
 - *Server Side KMS Key Identifier*
 - *Create via API*
 - *Java VFS Service*
 - *Write*
 - *Testing*
 - *Using S3 Connections*
 - *Uses of S3*
 - *Before you begin using S3*
 - *Secure access*
 - *Storing data in S3*
 - *Reading from sources in S3*
 - *Creating datasets*
 - *Writing results*
 - *Creating a new dataset from results*
 - *Purging files*
-

You can create connections to specific S3 buckets through the Trifacta application. These connections to S3 enable workspace users to read from and write to specific S3 buckets.

Simple Storage Service (S3) is an online data storage service provided by Amazon, which provides low-latency access through web services. For more information, see <https://aws.amazon.com/s3/>.

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Supported	Supported	Not supported

Prerequisites

Before you begin, please verify that your Trifacta® environment meets the following requirements:

- **Integration:** Your Trifacta instance is connected to a running environment supported by your product edition.
- **Multiple region:** Multiple S3 connections can be configured in different regions.
- Verify that `Enable S3 Connectivity` has been enabled in the *Workspace Settings Page*.
- Acquire the Access Key ID and Secret Key for the S3 bucket or buckets to which you are connecting. For more information on acquiring your key/secret combination, contact your S3 administrator.

Permissions

Access to S3 requires:

- Each user must have appropriate permissions to access S3.

NOTE: If a user does not have write permissions to the specified S3 bucket, publishing jobs to the bucket fail.

- To browse multiple buckets through a single S3 connection, additional permissions are required. See below.

Limitations

- Authentication using IAM roles is not supported.
- Automatic region detection in the create and edit connection is not supported.
- Publishing the output to multi-part files is not supported.

NOTE: For some file formats, like Parquet, multi-part files are the default output.

- Publishing the output using compression option is not supported for Trifacta Photon jobs.

Workaround: If you need to generate an output using compression to this S3 bucket, you can run the job on another running environment.

Create Connection

You can create additional S3 connections by the following method:

Create through application

You can create a S3 connection through the application.

Steps:

1. Login to the application.
2. In the left navigation bar, click the **Connections** icon.
3. In the Create Connection page, click the External Amazon S3 card.
4. Specify the connection properties:

Property	Description
DefaultBucket	(Optional) The default S3 bucket to which to connect. When the connection is first accessed for browsing, the contents of this bucket are displayed. If this value is not provided, then the list of available buckets based on the key/secret combination is displayed when browsing through the connection. NOTE: To see the list of available buckets, the connecting user must have the getBucketList permission. If that permission is not present and no default bucket is listed, then the user cannot browse S3.
Access Key ID	Access Key ID for the S3 connection.
Secret Key	Secret Key for the S3 connection.
Server Side Encryption	If server-side encryption has been enabled on your bucket, you can select the server-side encryption policy to use when writing to the bucket. SSE-S3 and SSE-KMS methods are supported. For more information, see http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html .

Server Side Kms key Id	When KMS encryption is enabled, you must specify the AWS KMS key ID to use for the server-side encryption. For more information, see "Server Side KMS Key Identifier" below.
------------------------	--

For more information on the other options, see *Create Connection Window*.

5. Click **Save**.

Server Side KMS Key Identifier

When KMS encryption is enabled, you must specify the AWS KMS key ID to use for the server-side encryption.

- Access to the key:
 - Access must be provided to the authenticating user.
 - The AWS IAM role must be assigned to this key.
- Encrypt/Decrypt permissions for the specified KMS key ID:
 - Permissions must be assigned to the authenticating user.
 - The AWS IAM role must be given these permissions.
 - For more information, see <https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html>.

The format for referencing this key is the following:

```
"arn:aws:kms:<regionId>:<acctId>:key/<keyId>"
```

You can use an AWS alias in the following formats. The format of the AWS-managed alias is the following:

```
"alias/aws/s3"
```

The format for a custom alias is the following:

```
"alias/<FSR>"
```

where:

<FSR> is the name of the alias for the entire key.

Create via API

For more information on the vendor and type information to use, see *Connection Types*.

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

API: *API Reference*

- Type: remotefile
- vendor: aws

Java VFS Service

The Java VFS Service has been modified to handle an optional connection ID, enabling S3 URLs with connection ID and credentials. The other connection details are fetched through the Trifacta application to create the required URL and configuration.

```
// sample URI
s3://bucket-name/path/to/object?connectionId=136

// sample java-vfs-service CURL request with s3
curl -H 'x-trifacta-person-workspace-id: 1' -X GET 'http://localhost:41917/vfsList?uri=s3://bucket-name/path/to/object?connectionId=136'
```

Write

You can publish results to your external S3 buckets. Configure an output destination to write to your external S3 bucket.

1. In Flow View, create or edit an output object.
 - a. To edit, right-click an output object. The object details are displayed in the Details panel.
2. In the Details panel, click **Edit**.
3. Modify the output destination to use the External S3 buckets connection.

NOTE: The `append` action is not supported when publishing to S3.

4. Navigate the bucket to select the appropriate location for the output. Specify the file as needed.
5. To save your changes, click **Update**.

For more information, see *Create Outputs*.

Testing

1. Import a dataset from External Amazon S3.
2. Add it to a flow and run a job, publishing results back to S3.

For more information, see *Verify Operations*.

Using S3 Connections

Uses of S3

Trifacta can use S3 for the following tasks:

1. **Enabled S3 Integration:** Trifacta has been configured to integrate with your S3 instance. For more information, see *S3 Access*.
2. **Creating Datasets from S3 Files:** You can read in source data stored in S3. An imported dataset may be a single S3 file or a folder of identically structured files. See *Reading from Sources in S3* below.
3. **Reading Datasets:** When creating a dataset, you can pull your data from a source in S3. See *Creating Datasets* below.
4. **Writing Results:** After a job has been executed, you can write the results back to S3. See *Writing Results* below.

In the Trifacta application, S3 is accessed through the S3 browser. See *S3 Browser*.

NOTE: When Trifacta application executes a job on a dataset, the source data is untouched. Results are written to a new location, so that no data is disturbed by the process.

Before you begin using S3

- **Access:** If you are using system-wide permissions, your administrator must configure access parameters for S3 locations. If you are using per-user permissions, this requirement does not apply. See [S3 Access](#).

Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Trifacta application.

- Your administrator should provide a writeable home output directory for you. This directory location is available through your user profile. See [Storage Config Page](#).

Secure access

Your administrator can grant access on a per-user basis or for the entire workspace.

Trifacta utilizes an S3 key and secret to access your S3 instance. These keys must enable read/write access to the appropriate directories in the S3 instance.

NOTE: If you disable or revoke your S3 access key, you must update the S3 keys for each user or for the entire system.

For more information, see [S3 Access](#).

Storing data in S3

Your administrator should provide raw data or locations and access for storing raw data within S3. All Trifacta users should have a clear understanding of the folder structure within S3 where each individual can read from and write results.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.
- The Trifacta application stores the results of each job in a separate folder in S3.

NOTE: Trifacta does not modify source data in S3. Source data stored in S3 is read without modification from source locations, and source data uploaded to Trifacta is stored in `/trifacta/uploads`.

Reading from sources in S3

You can create an imported dataset from one or more files stored in S3.

NOTE: To be able to import datasets from the base storage layer, your user account must include the `taAdmin` role.

NOTE: Import of glaciated objects is not supported.

Wildcards:

You can parameterize your input paths to import source files as part of the same imported dataset. For more information, see [Overview of Parameterization](#).

Folder selection:

When you select a folder in S3 to create your dataset, you select all files in the folder to be included.

Notes:

- This option selects all files in all sub-folders and bundles them into a single dataset. If your sub-folders contain separate datasets, you should be more specific in your folder selection.
- All files used in a single imported dataset must be of the same format and have the same structure. For example, you cannot mix and match CSV and JSON files if you are reading from a single directory.

When a folder is selected from S3, the following file types are ignored:

- `*_SUCCESS` and `*_FAILED` files, which may be present if the folder has been populated by the running environment.

NOTE: If you have a folder and file with the same name in S3, search only retrieves the file. You can still navigate to locate the folder.

Creating datasets

When creating a dataset, you can choose to read data in from a source stored from S3 or local file.

- S3 sources are not moved or changed.
- Local file sources are uploaded to `/trifacta/uploads` where they remain and are not changed.

Data may be individual files or all of the files in a folder. In the Import Data page, click the S3 tab. See *Import Data Page*.

Tip: Users can create secondary connections to specific S3 buckets. For more information, see *External S3 Connections*.

Writing results

When you run a job, you can specify the S3 bucket and file path where the generated results are written. By default, the output is generated in your default bucket and default output home directory.

- Each set of results must be stored in a separate folder within your S3 output home directory.
- For more information on your output home directory, see *Storage Config Page*.

If Trifacta installation is using S3, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Trifacta application can destroy other users' data. Please use the tools provided through the Trifacta application interface for managing uploads from S3.

NOTE: When writing files to S3, you may encounter an issue where the UI indicates that the job failed, but the output file or files have been written to S3. This issue may be caused when S3 does not report the files back to the application before the S3 consistency timeout has expired. For more information on raising this timeout setting, see *S3 Access*.

Creating a new dataset from results

As part of writing results, you can choose to create a new dataset, so that you can chain together data wrangling tasks.

NOTE: When you create a new dataset as part of your results, the file or files are written to the designated output location for your user account. Depending on how your permissions are configured, this location may not be accessible to other users.

Purging files

Other than temporary files, Trifacta does not remove any files that were generated or used by the platform, including:

- Uploaded datasets
- Generated samples
- Generated results

If you are concerned about data accumulation, you should create a bucket policy to periodically backup or purge directories in use. For more information, please see the S3 documentation.

Amazon Redshift Connections

Contents:

- *Prerequisites*
 - *Permissions*
 - *Limitations*
 - *Create Connection*
 - *Create through application*
 - *Connection URL*
 - *Example*
 - *Driver Information*
 - *Create via API*
 - *Troubleshooting*
 - *Testing*
 - *Using Redshift Connections*
 - *Uses of Redshift*
 - *Before you begin using Redshift*
 - *Secure access*
 - *Storing data in Redshift*
 - *Reading from Redshift*
 - *Writing to Redshift*
 - *Reference*
-

This section provides information on how to enable Amazon Redshift connectivity and create one or more connections to Amazon Redshift sources.

- Amazon Redshift is a hosted data warehouse available through Amazon Web Services. It is frequently used for hosting of datasets used by downstream analytic tools such as Tableau and Qlik. For more information, see <https://aws.amazon.com/redshift/>.
- When exporting results, you can choose to write to a Redshift database. See *Publishing Dialog*.

Supported Environments:

NOTE: S3 must be set as the base storage layer. See *Set Base Storage Layer*.

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Supported	Not supported
Write	Not supported	Supported	Not supported

Prerequisites

Before you begin, please verify that your Trifacta® environment meets the following requirements:

NOTE: If you are connecting to any relational source of data, such as Amazon Redshift or Oracle Database, you must add the Trifacta Service to your whitelist for those resources.

Tip: If the credentials used to connect to S3 do not provide access to Redshift, you can create an independent IAM role to provide access from Amazon Redshift to S3. If this separate role is available, the Amazon Redshift connection uses it instead. There may be security considerations.

1. **S3 base storage layer: Amazon Redshift** access requires use of S3 as the base storage layer, which must be enabled. See *Set Base Storage Layer*.
2. **Same region:** The Amazon Redshift cluster must be in the same region as the default S3 bucket.
3. **Integration:** Your Trifacta instance is connected to a running environment supported by your product edition.
4. **Deployment:** Trifacta platform is deployed either on-premises or in EC2.

Permissions

Access to Amazon Redshift requires:

- Each user is able to access S3
- S3 is the base storage layer

If the credentials used to connect to S3 do not provide access to Amazon Redshift, you can create an independent IAM role to provide access from Amazon Redshift to S3. If this separate role is available, the Amazon Redshift connection uses it instead.

NOTE: There may be security considerations with using an independent role to govern this capability.

Steps:

1. The IAM role must contain the required S3 permissions. See *Required AWS Account Permissions*.
2. The Amazon Redshift cluster should be assigned this IAM role. For more information, see <https://docs.aws.amazon.com/redshift/latest/mgmt/authorizing-redshift-service.html>.

Limitations

- You can publish any specific job once to Amazon Redshift through the export window. See *Publishing Dialog*.
- When publishing to Redshift through the Publishing dialog, output must be in Avro or JSON format. This limitation does not apply to direct writing to Amazon Redshift.
- Management of nulls:
 - Nulls are displayed as expected in the Trifacta application.
 - When Amazon Redshift jobs are run, the UNLOAD SQL command in Redshift converts all nulls to empty strings. Null values appear as empty strings in generated results, which can be confusing. This is a known issue with Amazon Redshift.
- No schema validation is performed as part of writing results to Redshift.
- Credentials and permissions are not validated when you are modifying the destination for a publishing job.
- For Redshift, no validation is performed to determine if the target is a view and is therefore not a supported target.

Create Connection

You can create Amazon Redshift connections through the following methods.

Tip: SSL connections are recommended. Details are below.

Create through application

Any user can create a Redshift connection through the application.

Steps:

1. Login to the application.
2. In the menu, click the Connections icon.
3. In the Create Connection page, click the **Amazon Redshift** connection card.
4. Specify the properties for your Amazon Redshift database connection:

Property	Description
Host	Hostname of the Amazon Redshift cluster NOTE: This value must be the full hostname of the cluster, which may include region information.
Port	Port number used to access the Amazon Redshift cluster. Default is 5439.
Connect String Options	Please insert any connection options as a string here. See below.
Database	The Amazon Redshift database to which to connect on the cluster
Credential Type	Options: Basic authentication with optional IAM role ARN: Basic authentication credentials specified in this window are used to connect to the Amazon Redshift database. Additional permissions may be governed by any ARN specified in the IAM role used for the account. Use this option if you are planning to specify a database username /password combination as part of the connection. IAM Role: Connection to Amazon Redshift is governed by the IAM role associated with the user's account.
Username	Username with which to connect to the Amazon Redshift database
Password	Password associated with the Amazon Redshift username
IAM Role ARN for Redshift /S3 connectivity	(Optional) You can specify an IAM role ARN that enables role-based connectivity between Amazon Redshift and the S3 bucket that is used as intermediate storage during Amazon Redshift bulk COPY/UNLOAD operations. Example: <code>arn:aws:iam::1234567890:role/MyRedshiftRole</code> For more information, see <i>Configure for EC2 Role-Based Authentication</i> .

For more information on the other options, see *Create Connection Window*.

5. Click **Save**.

Enable SSL connections

To enable SSL connections to Amazon Redshift, you must enable them first on your Amazon Redshift cluster. For more information, see <https://docs.aws.amazon.com/redshift/latest/mgmt/connecting-ssl-support.html>.

In your connection to Amazon Redshift, please add the following string to your Connect String Options:

```
;ssl=true
```

Save your changes.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:redshift://<host>:<port>/<database><connect-string-options>
```

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
;<prop1>=<val1>&<prop2>=<val2>;...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

Delimiters:

- `;` : any set of connect string options must begin and end with a semi-colon.
- `;` : all additional property names must be prefixed with a semi-colon.
- `=` : property names and values must be separated with an equal sign (=).

Example

Access through AWS key-secret

The following example connection URL uses an AWS key/secret combination (IAM user) to access Amazon Redshift:

```
jdbc:redshift:iam://<redshift_clustername:region_name>:<port_number>/<database_name>?  
AccessKeyId=<access_key_value>&SecretAccessKey=<secret_key_value>&DBUser=<database_user_name>
```

where:

- `<redshift_clustername>`: the name of the Amazon Redshift cluster
- `<region_name>`: region identifier where the cluster is located
- `<port_number>`: port number to use to access the cluster
- `<database_name>`: name of the Redshift database to which to connect
- `<access_key_value>`: identifier for the AWS key
- `<secret_key_value>`: identifier for the AWS secret
- `<database_user_name>`: user identifier for connecting to the database

Access through IAM role and temporary credentials

The following example connection URL uses an AWS/Key secret combination using temporary credentials:

```
jdbc:redshift:iam://<redshift_clustername:region_name>:<port_number>/<database_name>?  
AccessKeyID=<access_key_value>&SecretAccessKey=<secret_key_value>&SessionToken=<session_token>&DBUser=<databas  
e_user_name>
```

where:

- See previous.
- `<session_token>`: the AWS session token retrieved when using temporary credentials. The session token is requested by Trifacta when using AWS temporary credentials. For more information, see *Configure AWS Per-User Auth for Temporary Credentials*.

Driver Information

This connection uses the following driver:

- **Driver name:** `com.amazon.redshift.jdbc41.Driver`
- **Driver version:** `com.amazon.redshift:redshift-jdbc41-no-awssdk:1.2.45.1069`
- **Driver documentation:** <https://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html>

Create via API

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

API: *API Reference*

- Type: `redshift`
- vendor: `redshift`

Troubleshooting

For more information, see <https://docs.aws.amazon.com/redshift/latest/mgmt/troubleshooting-connections.html>.

Testing

Import a dataset from Amazon Redshift. Add it to a flow, and specify a publishing action. Run a job.

NOTE: When publishing to Amazon Redshift through the Publishing dialog, output must be in Avro or JSON format. This limitation does not apply to direct writing to Amazon Redshift.

For more information, see *Verify Operations*.

After you have run your job, you can publish the results to Amazon Redshift through the Job Details page. See *Publishing Dialog*.

Using Redshift Connections

Uses of Redshift

Trifacta can use Redshift for the following tasks:

1. Create datasets by reading from Redshift tables.

2. Write to Redshift tables with your job results.
3. Ad-hoc publication of data to Redshift.

Before you begin using Redshift

- **Enable S3 Sources:** Redshift integration requires the following:
 - S3 is set to the base storage layer.
 - For more information, see [S3 Access](#).
- **Read Access:** Your Redshift administrator must configure read permissions. Your administrator should provide a database for upload to your Redshift datastore.
- **Write Access:** You can write and publish jobs results to Redshift.

Secure access

SSL is required.

Storing data in Redshift

Your Redshift administrator should provide database access for storing datasets. Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: Trifacta does not modify source data in Redshift. Datasets sourced from Redshift are read without modification from their source locations.

Reading from Redshift

You can create a Trifacta dataset from a table or view stored in Redshift.

NOTE: The Redshift cluster must be in the same region as the default S3 bucket.

NOTE: If a Redshift connection has an invalid iamRoleArn, you can browse, import datasets, and open the data in the Transformer page. However, any jobs executed using this connection fail. If the iamRoleArn is invalid, the only samples that you can generate are Quick Random samples; other sampling jobs fail.

For more information, see [Database Browser](#).

Writing to Redshift

NOTE: You cannot publish to a Redshift database that is empty. The database must contain at least one table.

You can write back data to Redshift using one of the following methods:

- Job results can be written directly to Redshift as part of the normal job execution. Create a new publishing action to write to Redshift. See [Run Job Page](#).
- As needed, you can publish results to Redshift for previously executed jobs.

NOTE: You cannot re-publish results to Redshift if the original job published to Redshift. However, if the dataset was transformed but publication to Redshift failed, you can publish from the Publishing dialog.

NOTE: To publish to Redshift, the source results must be in Avro or JSON format.

- For more information on how data is converted to Redshift, see *Redshift Data Type Conversions*.

Data Validation issues:

- No validation is performed for the connection and any required permissions during job execution. So, you can be permitted to launch your job even if you do not have sufficient connectivity or permissions to access the data. The corresponding publish job fails at runtime.
- Prior to publication, no validation is performed on whether a target is a table or a view, so the job that was launched fails at runtime.

Reference

Supported Versions: n/a

Supported Environments:

NOTE: S3 must be set as the base storage layer. See *Set Base Storage Layer*.

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Supported	Not supported
Write	Not supported	Supported	Not supported

AWS Glue Connections

Contents:

- *Prerequisites*
 - *Limitations*
 - *Create Connection*
 - *Create through application*
 - *Connection URL*
 - *Driver Information*
 - *Troubleshooting*
 - *Use*
 - *Testing*
 - *Using AWS Glue Connections*
 - *Enable*
 - *Uses of Glue*
 - *Before you begin using Glue*
 - *Secure Access*
 - *Reading partitioned data*
 - *Storing data in Glue*
 - *Reading from Glue*
 - *Notes on reading from views using custom SQL*
 - *Writing to Glue*
 - *SQL Syntax*
 - *Reference*
-

This section describes how to create a connection to your AWS Glue Data Catalog.

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Not supported	Not supported	Not supported

Prerequisites

Before you create a connection, you must enable Trifacta to access AWS Glue. For more information, see *AWS Glue Access*.

Limitations

For more information, see "Supported Deployments" in *AWS Glue Access*.

Create Connection

You can create one or more connections to databases in your AWS Glue deployment.

Create through application

Any user can create an AWS Glue connection through the application.

Steps:

1. Login to the application.
2. In the menu, click **User menu > Preferences > Connections**.
3. In the Create Connection page, click the AWS Glue connection card.
4. Specify the properties for your AWS Glue connection. The following parameters are specific to AWS Glue connections:

Property	Description
EMR Master Node DNS	This DNS value can be retrieved from the EMR console.
Port	The port number through which to connect to the DNS master node
Connection String Options	No values are required here. Additional information is provided below.

For more information, see *Create Connection Window*.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:hive2://<host>:<port>/<database><connect-string-options>
```

where:

- `<database>` = name of the default database to which to connect. This value can be empty.

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
;<prop1>=<val1>;<prop2>=<val2>...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

Delimiters:

- `;` : any set of connect string options must begin with a semi-colon.
- `;` : sets of connect string options must separated by a semi-colon.
- `=` : property names and values must be separated with an equal sign (=).

Examples:

Trifacta may insert additional authentication properties as part of the connect string options.

Driver Information

This connection uses the following driver:

- **Driver name:** `org.apache.hive.jdbc.HiveDriver`
- **Driver version:** The driver depends on the version of EMR that is in use.

- **Driver documentation:**

<https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients#HiveServer2Clients-JDBC>

Troubleshooting

For more information, see <https://docs.aws.amazon.com/glue/latest/dg/troubleshooting-connection.html>.

Use

After the integration has been made between the platform and AWS Glue, you can import datasets.

- Import using custom SQL queries. For more information, see *Create Dataset with SQL*.

Testing

Import a dataset from AWS Glue. Add it to a flow, and run a job. Verify the results. For more information, see *Verify Operations*.

Using AWS Glue Connections

Enable

For more information, see *AWS Glue Access*.

Uses of Glue

The Trifacta platform can use Glue for the following tasks:

1. Create datasets by reading from Glue tables.

Before you begin using Glue

- **Read Access:** Your Glue administrator must configure read permissions to Glue databases.
- **Write Access:** Not supported.

Secure Access

For more information, see *Configure for AWS*.

Reading partitioned data

The Trifacta platform can read in partitioned tables. However, it cannot read individual partitions of partitioned tables.

Tip: If you are reading data from a partitioned table, one of your early recipe steps in the Transformer page should filter out the unneeded table data so that you are reading only the records of the individual partition.

Storing data in Glue

Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: The Trifacta platform does not modify source data in Glue. Datasets sourced from Glue are read without modification from their source locations.

Reading from Glue

You can create a Trifacta dataset from a table or view stored in Glue. For more information, see *Database Browser*.

Notes on reading from views using custom SQL

If you have enabled custom SQL and are reading data from a view, nested functions are written to a temporary filename, unless they are explicitly aliased.

Tip: If your custom SQL uses nested functions, you should create an explicit alias from the results. Otherwise, the job is likely to fail.

Problematic Example:

```
SELECT
  UPPER(`t1`.`column1`),
  TRIM(`t1`.`column2`),...
```

When these are read from a Glue view, the temporary column names are: `_c0`, `_c1`, etc. During job execution, Spark ignores the `column1` and `column2` reference.

Improved Example:

```
SELECT
  UPPER(`t1`.`column1`) as col1,
  TRIM(`t1`.`column2`) as col2,...
```

In this improved example, the two Glue view columns are aliased to the explicit column names, which are correctly interpreted and used by the Spark running environment during job execution.

Writing to Glue

Not supported.

SQL Syntax

The following syntax requirements apply to this connection.

Object delimiter: backtick

Example syntax:

```
SELECT `column1`,`column2` FROM `databaseName`.`tableName`;
```

For more information on SQL in general, see *Supported SQL Syntax*.

Reference

Supported Versions: n/a

Supported Environments:

NOTE: S3 must be set as the base storage layer, and the platform must be integrated with EMR. See *Set Base Storage Layer*.

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Not supported	Not supported	Not supported

Snowflake Connections

Contents:

- *Prerequisites*
 - *Prerequisites for OAuth 2.0*
 - *Limitations*
 - *Create Connection*
 - *Create through application*
 - *Connection URL*
 - *Driver Information*
 - *Create via API*
 - *Troubleshooting*
 - *Testing*
 - *Using Snowflake Connections*
 - *Uses of Snowflake*
 - *Before you begin using Snowflake*
 - *Secure access*
 - *Storing data in Snowflake*
 - *Reading from Snowflake*
 - *Writing to Snowflake*
-

This section describes how to create a connection to your Snowflake datawarehouse.

- Snowflake is an S3-based data warehouse service hosted in the cloud. Auto-scaling, automatic failover, and other features simplify the deployment and management of your enterprise's data warehouse. For more information, see <https://www.snowflake.com>.

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Supported	Not supported
Write	Not supported	Supported	Not supported

Prerequisites

- **S3 base storage layer:** Snowflake access requires installation of Trifacta software in the AWS infrastructure and use of S3 as the base storage layer, which must be enabled. See *Set Base Storage Layer*.
- **Integration:** Your Trifacta instance is connected to an EMR cluster. See *Configure for EMR*.
- **Deployment:** Trifacta platform is deployed in EC2.
- Integration with Snowflake requires deployment of the Trifacta platform within a customer-managed AWS infrastructure. For more information, see *Snowflake Access*.
- **PUBLIC schema:** If you do not create an external staging database:
 - A `PUBLIC` schema is required in your default database.
 - If you do not provide a stage database, then a temporary stage is created for you under the `PUBLIC` schema in the default database.

- **S3 bucket:** The user-created stage must point to the same S3 bucket as the default bucket in use by Trifacta.
- **Same region:** The Snowflake cluster must be in the same region as the default S3 bucket.
- **IAM role requirements:** If you are accessing AWS and Snowflake using IAM roles, please verify that the appropriate permissions have been assigned to the role to access Snowflake and its backing S3 buckets. For more information, see *Required AWS Account Permissions*.
- **Staging database:** Snowflake supports the use of a stage for reading and writing data to S3 during job executions.

NOTE: If a stage is not deployed, then the user must have write permissions to the default database, which is used instead for staging your data in Snowflake. These permissions must be included in the AWS credentials applied to the user account.

For more information, see *Snowflake Access*.

Prerequisites for OAuth 2.0

If you are connecting to your Snowflake deployment using OAuth 2.0 authentication, additional configuration is required:

- OAuth 2.0 must be enabled and configured for use in the product. For more information, see *Enable OAuth 2.0 Authentication*.
- OAuth 2.0 requirements:
 - Create a security integration in your Snowflake deployment.
 - Create an OAuth 2.0 client in the Trifacta application that connects using the security integration.
 - For more information, see *OAuth 2.0 for Snowflake*.

Limitations

- You cannot perform ad-hoc publication to Snowflake.
- SSO connections are not supported.
- To ingest data from a Snowflake table, one of the following must be enabled:
 - A named stage must be created for the table. For more information, see the Snowflake documentation.
 - Snowflake must be permitted to create a temporary stage, which requires:
 - Write permissions on the table's database, and
 - A schema named PUBLIC must exist and be accessible.
- No schema validation is performed as part of writing results to Snowflake.
- Credentials and permissions are not validated when you are modifying the destination for a publishing job.
- For Snowflake, no validation is performed to determine if the target is a view and is therefore not a supported target.

Create Connection

You can create Snowflake connections through the following methods.

Create through application

Any user can create a Snowflake connection through the application.

Steps:

1. Login to the application.

2. In the left nav bar, click the Connections icon.
3. In the Create Connection page, click the Snowflake connection card.
4. Specify the properties for your Snowflake database connection. The following parameters are specific to Snowflake connections:

NOTE: In Snowflake connections, property values are case-sensitive. Snowflake-related locations are typically specified in capital letters.

Property	Description
Account Name	<p>Snowflake account to use. Suppose your hostname is the following:</p> <pre>mycompany.snowflakecomputing.com</pre> <p>Your account name is the following:</p> <pre>mycompany</pre> <p>NOTE: Your full account name might include additional segments that identify the region and cloud platform where your account is hosted.</p>
Warehouse	<p>The name of the warehouse to use when connected. This value can be an empty string.</p> <p>If specified, the warehouse should be an existing warehouse for which the default role has privileges.</p>
Stage	<p>If you have deployed a Snowflake stage for managing file conversion to tables, you can enter its name here. A stage is a database object that points to an external location on S3. It must be an external stage containing access credentials.</p> <p>If a stage is used, then this value is typically the schema and the name of the stage. Example value:</p> <pre>MY_SCHEMA.MY_STAGE</pre> <p>If a stage is not specified, a temporary stage is created using the current user's AWS credentials.</p> <p>NOTE: Without a defined stage, you must have write permissions to the database from which you import. This database is used to create the temporary stage.</p> <p>For more information on stages, see https://docs.snowflake.net/manuals/sql-reference/sql/create-stage.html.</p>
Credential Type	<p>Select the type of credentials to provide with the connection:</p> <ul style="list-style-type: none"> • Basic - username and password are used by the connection to authenticate to Snowflake. • OAuth 2.0 - use OAuth 2.0 client connect to Snowflake. The client must already be defined in the Trifacta application and then selected in the connection configuration. <p>NOTE: After you have specified the connection to use OAuth 2.0, click Authenticate to validate the connection with the target datastore. If you have modified the connection, click Re-authenticate to validate the new connection definition. You must re-authenticate if you receive an expired tokens message. For more information, see <i>Enable OAuth 2.0 Authentication</i>.</p> <p>For more information, see <i>OAuth 2.0 for Snowflake</i>.</p>
Database for Stage	<p>(optional) If you are using a Snowflake stage, you can specify a database other than the default one to host the stage.</p>

NOTE: If you are creating a read-only connection to Snowflake, this field is required. The accessing user must have write permission to the specified database.

If no value is specified, then your stage must be in the default database.

For more information, see *Create Connection Window*.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:snowflake://<account_name>.snowflakecomputing.com/?db=<database>&warehouse=<warehouse><connect-string-options>
```

where:

- `<database>` = name of the default database to which to connect. This value can be empty.

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
&<prop1>=<val1>&<prop2>=<val2>...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

Delimiters:

- `&` : any set of connect string options must begin with an ampersand (&).
- `=` : property names and values must be separated with an equal sign (=).

Disable SSL connections

By default, connections to Snowflake use SSL. To disable, please add the following string to your Connect String Options:

```
:ssl=false
```

Connect through proxy

If you require connection to Snowflake through a proxy server, additional Connect String Options are required. For more information, see <https://docs.snowflake.net/manuals/user-guide/jdbc-configure.html#specifying-a-proxy-server-in-the-jdbc-connection-string>

Driver Information

This connection uses the following driver:

- **Driver name:** `net.snowflake.client.jdbc.SnowflakeDriver`
- **Driver version:** `net.snowflake:snowflake-jdbc:3.8.5`
- **Driver documentation:** <https://docs.snowflake.com/en/user-guide/jdbc.html>

Create via API

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnectionAPI>:
API Reference

- Type: snowflake
- vendor: snowflake

Troubleshooting

Error Message	Description
Null values in some columns for all rows	When there are spaces/special characters in columns names, null values can be inserted for all rows in the column. The workaround is to remove any special characters and spaces from column names.

Testing

Import a dataset from Snowflake. Add it to a flow, and specify a publishing action back to Snowflake. Run a job. For more information, see *Verify Operations*.

Using Snowflake Connections

Uses of Snowflake

Trifacta can use Snowflake for the following tasks:

1. Create datasets by reading from Snowflake tables.
2. Write to Snowflake tables with your job results.

Before you begin using Snowflake

- **Enable S3 Sources:** Snowflake integration requires the following:
 - Installation of the product on a customer-managed AWS infrastructure.
 - S3 is set to the base storage layer.
 - For more information, see *Snowflake Access*.
- **Read Access:** Your Snowflake administrator must configure read permissions. Your administrator should provide a database for upload to your Snowflake data warehouse.
 - **Read-only Access:** If you are creating a read-only connection to Snowflake, you must provide a database for staging. The accessing user must have write permission to the specified database.
- **Write Access:** You can write and publish jobs results to Snowflake.

Secure access

SSL is the default connection method.

Storing data in Snowflake

Your Snowflake administrator should provide database access for storing datasets. Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: Trifacta does not modify source data in Snowflake. Datasets sourced from Snowflake are read without modification from their source locations.

Reading from Snowflake

You can create a Trifacta dataset from a table stored in Snowflake.

NOTE: The Snowflake cluster must be in the same region as the default S3 bucket.

Writing to Snowflake

You can write back data to Snowflake using one of the following methods:

- Job results can be written directly to Snowflake as part of the normal job execution. Create a new publishing action to write to Snowflake. See *Run Job Page*.
- For more information on how data is converted to Snowflake, see *Snowflake Data Type Conversions*.

Data Validation issues:

- No validation is performed for the connection and any required permissions during job execution. So, you can be permitted to launch your job even if you do not have sufficient connectivity or permissions to access the data. The corresponding publish job fails at runtime.
- Prior to publication, no validation is performed on whether a target is a table or a view, so the job that was launched fails at runtime.

Azure SQL Database Connections

Contents:

- *Limitations*
- *Prerequisites*
- *Configure*
 - *Configure for SSO*
- *Use*
- *Data Conversion*

You can create a connection to a Microsoft Azure SQL Database from Trifacta®. This section describes how to create connections of this type.

- This connection type supports data ingestion into ADLS/WASB. When large volumes of data are read from an Azure SQL Database during job execution, the data is stored in a temporary location in ADLS/WASB. After the job has been executed, the data is removed from the datastore. This process is transparent to the user.
- For more information on Azure SQL Database, see <https://azure.microsoft.com/en-us/services/sql-database/>.

NOTE: This database connection is a specialized version of a SQL Server connection.

NOTE: For Azure deployments, some additional configuration properties must be applied. See *Configure for Azure*.

Supported Versions: Azure SQL Database version 12 (other versions are not supported)

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Not supported	Supported
Write	Supported	Not supported	Supported

Limitations

- Connections of this type cannot be created via API.

Prerequisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Configure

- For additional details on creating an Azure SQL Database connection, see *Relational Access*.

Please create an Azure SQL Database connection and then specify the following properties with the listed values:

Property	Description
----------	-------------

Host	<p>Enter your hostname. Example:</p> <pre>testsql.database.windows.net</pre>
Port	Set this value to 1433.
Connect String options	<p>Please insert the following as a single string (no line breaks):</p> <pre>;encrypt=true;trustServerCertificate=false; hostNameInCertificate=*.database.windows.net;loginTimeout=30;</pre> <div> <p>Tip: If you have access to the Azure SQL Database through the Azure SQL Database Portal, please copy the Connect String from that configuration. You may omit the username and password from that version of the string.</p> </div>
Database	(optional) Name of the Azure SQL Database to which you are connecting.
User Name	(for basic Credential Type) Username to use to connect to the database.
Password	(for basic Credential Type) Password associated with the above username.
Credential Type	<ul style="list-style-type: none"> • <code>basic</code> - Specify username and password as part of the connection • <code>Azure Token SSO</code> - Use the SSO principal of the user creating the connection to authenticate to the Azure SQL Database. Additional configuration is required. See <i>Enable SSO for Azure Relational Connections</i>.
Default Column Data Type Inference	Set to <code>disabled</code> to prevent Trifacta from applying its own type inference to each column on import. The default value is <code>enabled</code> .

Configure for SSO

If you have enabled Azure AD SSO integration for the Trifacta platform, you can create SSO connections to Azure relational databases. See *Enable SSO for Azure Relational Connections*.

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *SQL Server Data Type Conversions*.

Microsoft SQL Data Warehouse Connections

Contents:

- Overview
 - Table types
 - SQL pool types
- Limitations
- Prerequisites
- Azure Synapse Analytics (Formerly Microsoft SQL DW) permissions
- Azure Synapse Analytics (Formerly Microsoft SQL DW) External Data Source Name
- Configure
 - Configure for SSO
- Use
- Data Conversion

This section describes how to create connections to Azure® Synapse Analytics (Formerly Microsoft® SQL DW)®.

Tip: This connection is now known as Azure Synapse Analytics.

Supported Versions: n/a

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported

Overview

Table types

Azure Synapse Analytics (Formerly Microsoft SQL DW) can interact with the following table types:

Table type	Description
Managed table	Managed tables are database tables that are specifically defined within the database server. Read and write are supported.
External table	External tables are references to files on the backend storage layer on top of which is a database schema. The table is defined by reading and writing through the database schema to the underlying file storage. <div>NOTE: When publishing to an external table, the output file type is Parquet.</div> <div>NOTE: When publishing to an external table under ADLS user mode, the system credentials are used to write to the storage location and must have the appropriate permissions.</div>

SQL pool types

Azure Synapse Analytics (Formerly Microsoft SQL DW) supports two different SQL pooling methods through which you connect to your data managed by the database server.

Tip: The type of SQL pooling in use is determined by the URL that you use to connect to Azure Synapse Analytics (Formerly Microsoft SQL DW). URLs with `ondemand` in them are for serverless SQL pool connections.

Dedicated SQL pool

These connections utilize a fixed and dedicated set of SQL pool resources. The admin user can define the size and availability of these resources for performing work.

This connection requires more permissions. You must also specify an External Datasource Name. See below.

Tip: Spark-based jobs that read or write through your Azure Synapse Analytics (Formerly Microsoft SQL DW) connection leverage PolyBase for faster performance.

Supported table types: Managed tables, external tables

Serverless SQL pool

These connections specify the SQL pool resources based on the size of the job. In theory, these connection types can scale infinitely for jobs of any size.

Tip: This connection requires fewer permissions on the data warehouse and its databases but is less performant. The URL for these connections always contain `ondemand`.

Supported table types: External tables only

Limitations

- Azure Synapse Analytics (Formerly Microsoft SQL DW) connections are available only if you have deployed the Trifacta® platform onto Azure.
- SSL connections to Azure Synapse Analytics (Formerly Microsoft SQL DW) are required.

NOTE: In this release, this connection cannot be created through the APIs. Please create connections of this type through the application.

NOTE: Under Azure SSO, write operations are not supported through Azure Synapse Analytics (Formerly Microsoft SQL DW) connections.

- JSON files cannot be read in through this connection type.
- For custom SQL and file formats other than CSV and Parquet, data is read through CETAS (create external table and select).
 - In some cases, reading from CETAS tables may exceed 30 minutes, which is the read limit imposed by Azure. These jobs time out.
 - In timeout situations, you may be able to fall back to a direct JDBC read of these sources.
- When publishing to an external table, the output file type is always Parquet.

Prerequisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Azure Synapse Analytics (Formerly Microsoft SQL DW) permissions

- **Read:** The authenticating DB user must have read permissions to any Azure Synapse Analytics (Formerly Microsoft SQL DW), schemas and tables to which the user should have access.
- **Write:** In addition to the above, the authenticating DB user must have the following permissions:

```
CREATE TABLE**
ALTER ANY SCHEMA
ALTER ANY EXTERNAL DATA SOURCE
ALTER ANY EXTERNAL FILE FORMAT
```

- The authenticating DB user must also have read access to the external data source.

Azure Synapse Analytics (Formerly Microsoft SQL DW) External Data Source Name

When specifying a connection to external tables, you can provide an External Data Source Name value as part of the connection definition. The External Data Source enables publishing and support for large-scale data ingestion.

When the External Data Source is provided:

- CETAS (create external table and select) is used for reading in data.

If the External Data Source is not provided:

- JDBC read is used for reading in data.
- The connection is read-only.
- The connection must be to a set of managed tables.
- The native ingestion of the Trifacta platform is used.

Requirements:

- The external data source must be created by the database admin on the default database defined in the connection. For more information, see <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-data-source-transact-sql?view=azure-sqldw-latest&tabs=dedicated>
- The External Data Source must point to the same storage location as the base storage layer for the Trifacta platform. For example, if the base storage layer is WASB, the External Datasource must point to the same storage account defined in Trifacta configuration. If this configuration is incorrect, then publishing and ingestion of data fail.
- For more information on privileges required for the authenticating DB user, see <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-table-transact-sql>.

Configure

To create this connection, see *Connections Page*.

For additional details on creating a relational connection, see *Relational Access*.

Please create a connection of this type and modify the following properties with the listed values:

--	--

Property	Description
Host	<p>Enter your hostname. Example:</p> <pre>testsql.database.windows.net</pre> <p>Tip: If your Host value contains <code>ondemand</code>, then you are using serverless SQL pools.</p>
Port	Set this value to 1433.
Database	Set this value to the default database name.
External Data Source Name	For external table connections, you must provide an External Data Source. See above for details.
Connect String options	Include any options required for your environment:
User Name	Username to use to connect to the database.
Password	Password associated with the above username.
Credential Type	<ul style="list-style-type: none"> <code>basic</code> - Specify username and password as part of the connection <code>Azure Token SSO</code> - Use the SSO principal of the user creating the connection to authenticate to the SQL Server database. Additional configuration is required. See <i>Enable SSO for Azure Relational Connections</i>.
Default Column Data Type Inference	Set to <code>disabled</code> to prevent the Trifacta platform from applying its own type inference to each column on import. The default value is <code>enabled</code> .

Configure for SSO

If you have enabled Azure AD SSO integration for the Trifacta platform, you can create SSO connections to Azure relational databases.

NOTE: When Azure AD SSO is enabled, write operations to Azure Synapse Analytics (Formerly Microsoft SQL DW) are not supported.

See *Enable SSO for Azure Relational Connections*.

Use

For more information on locating data, see *Database Browser*.

For more information, see *Using SQL DW*.

For more information on defining output objects, see *Microsoft SQL Data Warehouse Table Settings*.

Data Conversion

For more information on how values are converted during input and output with this database, see *SQL DW Data Type Conversions*.

Databricks Tables Connections

Contents:

- *Limitations*
 - *Prerequisites*
 - *Insert Databricks Access Token*
 - *Enable*
 - *Create Connection*
 - *Connection URL*
 - *Driver Information*
 - *Data Conversion*
 - *Create via API*
 - *Troubleshooting*
 - *Failure when importing wide Databricks Tables table*
 - *Using Databricks Table Connections*
 - *Uses of Databricks tables*
 - *Before you begin using Databricks tables*
 - *Storing data in Databricks tables*
 - *Reading from Databricks Tables*
 - *Writing to Databricks Tables*
 - *Ad-hoc Publishing to Databricks Tables*
 - *Reference*
-

You can create a connection to Databricks Tables from the Trifacta platform. This section describes how to create connections of this type.

- Databricks Tables provides a JDBC-based interface for reading and writing datasets in ADLS or WASB. Using the underlying JDBC connection, you can access your ADLS or WASB data like a relational datastore, run jobs against it, and write results back to the datastore as JDBC tables.
- Your connection to Databricks Tables leverages the SSO authentication that is native to Databricks.
 - For more information on Azure Databricks Tables, see <https://docs.microsoft.com/en-us/azure/databricks/data/tables>.

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Supported	Supported
Write	Not supported	Supported	Supported

Limitations

- Ad-hoc publishing of generated results to Databricks Tables is not supported.
- Integration with Kerberos or secure impersonation is not supported.
- Some table types and publishing actions are not supported.
- Access to external Hive metastores is not supported.

Prerequisites

- **Azure:** The Trifacta platform must be installed on Azure and integrated with an Azure Databricks cluster.
 - See *Install for Azure*.
 - See *Configure for Azure Databricks*.
-

NOTE: For job execution on Spark, the connection must use the Spark instance on the Azure Databricks cluster. No other Spark instance is supported. You can run jobs from this connection through the Photon running environment. For more information, see *Running Environment Options*.

- **AWS:** The Trifacta platform must be installed on AWS and integrated with an AWS Databricks cluster.
 - See *Install for AWS*.
 - See *Configure for AWS Databricks*.

NOTE: For job execution on Spark, the connection must use the Spark instance on the AWS Databricks cluster. No other Spark instance is supported. You can run jobs from this connection through the Photon running environment. For more information, see *Running Environment Options*.

- This connection interacts with Databricks Tables through the Hive metastore that has been installed in the Databricks cluster.

NOTE: External Hive metastores are not supported.

Insert Databricks Access Token

Each user must insert a Databricks Personal Access Token into the user profile. For more information, see *Databricks Settings Page*.

Enable

To enable Databricks Tables connections, please complete the following:

NOTE: Typically, you need only one connection to Databricks Tables, although you can create multiple connections.

NOTE: This connection is created with SSL automatically enabled.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `true`:

```
"feature.databricks.connection.enabled": true,
```

3. To allow for direct publishing of job results to Databricks tables from the Run Job page, you must enable the following parameters. For more information on these settings, see *Databricks Tables Table Settings*.

Parameter	Description
feature.databricks.enableDeltaTableWrites	Set this value to <code>true</code> to enable users to choose to write generated results to Databricks delta tables from the Run Job page.

feature.databricks. enableExternalTableWrites	Set this value to <code>true</code> to enable users to choose to write generated results to Databricks external tables from the Run Job page.
--	---

4. Save your changes and restart the platform.

Create Connection

This connection can also be created via API. For details on values to use when creating via API, see *Connection Types*.

Please create a Databricks connection and then specify the following properties with the listed values:

NOTE: Host and port number connection information is taken from Databricks and does not need to be re-entered here.

- See *Configure for Azure Databricks*.
- See *Configure for AWS Databricks*.

Property	Description
Connect String options	Please insert any connection string options that you need. Connect String options are not required for this connection.
Test Connection	Click this button to test the specified connection.
Default Column Data Type Inference	Set to <code>disabled</code> to prevent the Trifacta platform from applying its own type inference to each column on import. The default value is <code>enabled</code> .

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:spark://<host>:<port>/<database><connect-string-options>
```

The Connection URL is mostly built up automatically using cluster configuration for the platform.

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
;<prop1>=<val1>;<prop2>=<val2>...
```

where:

- `<prop>` : the name of the property
- `<val>` : the value for the property

delimiters:

- `;` : any set of connect string options must begin and end with a semi-colon.
 - A semi-colon can be omitted from the end of the connect string options.
- `=` : property names and values must be separated with an equal sign (=).

Use HTTP

To enable the use of the HTTP protocol, specify the following in the connect string options:

```
;transportMode=http;
```

Use SSL

To enable the use of SSL for the connection, specify the following in the connect string options:

```
;ssl=1;
```

HTTP Path

When HTTP is enabled, you can specify the path as a connect string option:

```
;httpPath=sql/protocolv1/o/0/xxxx-xxxxxx-xxxxxxx;
```

Authentication

You can specify a Databricks personal access token to use when authenticating to the database using the following connect string options.

```
;AuthMech=3;UID=token;PWD=<Databricks-personal-access-token>
```

where:

- `<Databricks-personal-access-token>` = the personal access token of the user who is connecting to the database.

Driver Information

This connection uses the following driver:

- **Driver name:** `com.simba.spark.jdbc41.Driver`
- **Driver version:** `com.simba.jdbc:SparkJDBC41:2.6.11.1014`
- **Driver documentation:** <https://docs.databricks.com/integrations/bi/jdbc-odbc-bi.html>

Data Conversion

For more information on how values are converted during input and output with this database, see *Databricks Tables Data Type Conversions*.

Create via API

API: *API Reference*

- Type: `jdbc`
- Vendor: `databricks`

Troubleshooting

For more information on error messages for this connection type, see <https://kb.databricks.com/bi/jdbc-odbc-troubleshooting.html>.

Failure when importing wide Databricks Tables table

If you are attempting to import a table containing a large number of columns (>200), you may encounter an error message similar to the following:

```
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 408.0 failed 4 times, most recent failure: Lost task 0.3 in stage 408.0 (TID 1342, 10.139.64.11, executor 11): org.apache.spark.SparkException: Kryo serialization failed: Buffer overflow. Available: 0, required: 1426050. To avoid this, increase spark.kryoserializer.buffer.max value.
```

The problem is that the serializer ran out of memory.

Solution:

To address this issue, you can increase the Kryoserializer buffer size.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the `spark.props` section and add the following setting. Modify 2000 (2GB) depending on whether your import is successful:

```
"spark.kryoserializer.buffer.max.mb": "2000"
```

3. Save your changes and restart the platform.
4. Attempt to import the dataset again. If it fails, you can try incrementally raising the above value.

For more information on passing property values into Spark, see *Configure for Spark*.

Using Databricks Table Connections

Uses of Databricks tables

The Trifacta platform can use Databricks Tables for the following tasks:

1. Create datasets by reading from Databricks Tables tables.
2. Write data to Databricks Tables.

Table Type	Support	Notes
Databricks managed tables	Read /Write	
Delta tables	Read /Write	NOTE: Versioning and rollback of Delta tables is not supported within the Trifacta platform . The latest version is always used. You must use external tools to manage versioning and rollback.
External tables	Read /Write	NOTE: When writing to an external table the TRUNCATE and DROP publishing actions are not supported.
Databricks unmanaged tables	Read /Write	
Delta Tables (managed and unmanaged tables)	Read /Write	
Partitioned tables	Read	

The underlying format for Databricks Tables is Parquet.

Before you begin using Databricks tables

- **Databricks Tables deployment:** Your Trifacta administrator must enable use of Databricks Tables.
- **Databricks Personal Access Token:** You must acquire and save a Databricks Personal Access Token into your Trifacta account. For more information, see *Databricks Settings Page*.

Storing data in Databricks tables

NOTE: The Trifacta platform does not modify source data in Databricks Tables. Datasets sourced from Databricks Tables are read without modification from their source locations.

Reading from Databricks Tables

You can create a Trifacta dataset from a table or view stored in Databricks Tables.

- Read support is also available for Databricks Delta Lake.

NOTE: Custom SQL queries are supported. Multi-statement custom SQL is not supported for Databricks Tables. Custom SQL queries must be a single `SELECT` statement. For more information, see *Create Dataset with SQL*.

For more information on how data types are imported from Databricks Tables, see *Databricks Tables Data Type Conversions*.

Writing to Databricks Tables

You can write data back to Databricks Tables using one of the following methods:

- Job results can be written directly to Databricks Tables as part of the normal job execution.
 - Data is written as a managed table to DBFS in Parquet format.
 - Create a new publishing action to write to Databricks Tables. See *Run Job Page*.
- For more information on how data is converted to Databricks Tables, see *Databricks Tables Data Type Conversions*.

Ad-hoc Publishing to Databricks Tables

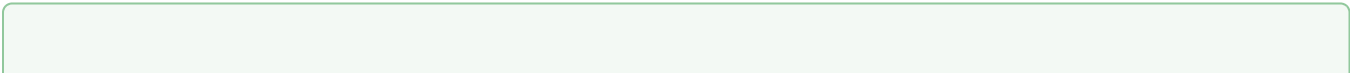
Not supported.

Reference

Supported Versions: n/a

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Supported	Supported
Write	Not supported	Supported	Supported



Tip: It's easier to create a connection of this type through the UI. Typically, only one connection is needed.

SFTP Connections

Contents:

- *Limitations*
 - *Prerequisites*
 - *SSH Keys*
 - *Whitelist SFTP server*
 - *Enable*
 - *Configure file storage protocols and locations*
 - *Enforce authentication methods*
 - *Java VFS service*
 - *Create Connection*
 - *Create through application*
 - *Create through APIs*
-

You can create connections to SFTP servers to upload your datasets to the Trifacta® application.

Linux- and Windows-based SFTP servers are supported.

Jobs can be executed from SFTP sources on the following running environments:

- HDFS-based Spark
- Azure Databricks
- Trifacta Photon
- Spark on EMR

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Limitations

- Files and folders with spaces or special characters in them cannot be used. For example, a file or folder on the SFTP server with a hashtag (#) in it cannot be used for data.
 - Files and folders whose names begin with underscore (_) are not visible.
- Ingest of over 500 files through SFTP at one time is not supported.
- You cannot run Spark jobs using Avro or Parquet sources uploaded via SFTP.
- You cannot publish compressed Snappy files to SFTP destinations.
- You cannot publish Hyper format to SFTP destinations.

Prerequisites

- Acquire user credentials to access the SFTP server. You can use username/password credentials or SSH keys. See below.
- Verify that the credentials can access the proper locations on the server where your data is stored. Initial directory of the user account must be accessible.

SSH Keys

If preferred, you can use SSH keys to for authentication to the SFTP server.

NOTE: SSH keys must be private RSA keys. If you have OpenSSH keys, you can use the `ssh-keygen` utility to convert them to private RSA keys.

Whitelist SFTP server

If you are running jobs on EMR or Azure Databricks, you must add the SFTP server to the whitelist of IPs that are permitted to communicate with the cluster. For more information, please see the documentation that is provided with your software distribution.

You must also add the SFTP server to the whitelist of file storage systems. Details are below.

Enable

By default, this connection type is automatically enabled for use.

NOTE: You must provide the protocol identifier and storage locations for the SFTP server. See below.

Configure file storage protocols and locations

The Trifacta platform must be provided the list of protocols and locations for accessing SFTP.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameters and set their values according to the table below:

```
"fileStorage.whitelist": ["sftp"],  
"fileStorage.defaultBaseUri": ["sftp:///"],
```

Parameter	Description
filestorage.whitelist	A comma-separated list of protocols that are permitted to access SFTP. <div>NOTE: The protocol identifier <code>"sftp"</code> must be included in this list.</div>
filestorage.defaultBaseUri	For each supported protocol, this parameter must contain a top-level path to the location where platform files can be stored. These files include uploads, samples, and temporary storage used during job execution.

NOTE: A separate base URI is required for each supported protocol. You may only have one base URI for each protocol.

NOTE: For SFTP, three slashes at the end are required, as the third one is the end of the path value. This value is used as the base URI for all SFTP connections created in Trifacta.

Example:

```
sftp:///
```

The above example is the most common example, as it is used as the base URI for all SFTP connections that you create. If you add a server value to the above URI, you limit all SFTP connections that you create to that specified server.

3. Save your changes and restart the platform.

Enforce authentication methods

By default, the Trifacta application enables use of two different authentication mechanisms:

- Basic - use a password to access the SFTP server
- SSHKey - use a public SSHKey and password to access the SFTP server

Along with basic and SSH key, the SFTP servers in your environment may be configured with other authentication methods, and those methods sometimes take precedence. As a result, when using default authentication methods, SFTP connections from the Trifacta platform can fail to connect to the SFTP server.

To eliminate these issues, you can configure the Trifacta application to enforce usage of one of the following authentication schemes. These schemes are passed to the SFTP server during connection time, which forces the server to use the appropriate method of authentication. When the following parameter is specified, SFTP connections can be configured using the listed methods and should work for connecting to the server.

NOTE: Enforcement applies to connections created via the APIs as well. After configuration, please be sure to use one of the enforced authentication methods when configuring your SFTP connections through the application or the APIs.

Steps:

1. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. Some of these settings may not be available through the *Admin Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter in the configuration file:

```
"batchserver.workers.filewriter.hadoopConfig.sftp.PreferredAuthentications"
```

3. Set the parameter value according to the following:

Preferred authentication method	Parameter value	Description
Basic	"password"	Basic password authentication method is used to connect to the SFTP server. <div>NOTE: You must configure your SFTP server connection in the platform to use the Basic method.</div>
SSHKey	"publickey"	SSH Key authentication method is used.

		NOTE: You must configure your SFTP server connection in the platform to use the SSHKey method.
both	"publickey, password"	Both methods of authentication are supported.

4. Save your changes and restart the platform.

Java VFS service

Use of SFTP connections requires the Java VFS service.

NOTE: This service is enabled by default.

For more information on configuring this service, see *Configure Java VFS Service*.

Create Connection

Create through application

You can create a SFTP connection through the Trifacta application.

Steps:

1. In the left nav bar, select the Connections icon. See *Connections Page*.
2. In the Connections page, click **Create Connection**. See *Create Connection Window*.
3. In the Create Connection window, click the SFTP connection card.
4. Specify the properties for your SFTP server.

Property	Description
Host	The hostname of the FTP server to which you are connecting. Do not include any protocol identifier (<code>sftp://</code>).
Port	The port number to use to connect to the server. Default port number is 22.
Credential Type	Select one of the following: basic - authenticate via username and password SSH Key - authenticate via username and SSH key
User Name	The username to use to connect.
Password	(Basic credential type) The password associated with the username.
SSH Key	(SSH Key credential type) The SSH key that applies to the username.
Test Connection	Click this button to test the connection that you have specified.
Default Directory	Absolute path on the SFTP server where users of the connection can begin browsing.
Block Size (Bytes)	Fetch size in bytes for each read from the SFTP server. NOTE: Raising this value may increase speed of read operations. However, if it is raised too high, resources can become overwhelmed, and the read can fail.
Connection Name	The name of the connection as you want it to appear in the application.

Description	This description is displayed in the application.
-------------	---

For more information, see *Create Connection Window*.

5. Click **Save**.

Create through APIs

- Type: `jdbc`
- Vendor: `sftp`

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

REST API Connections

Contents:

- *Limitations*
 - *Prerequisites*
 - *Enable connections on the same cluster*
 - *Configure*
 - *Authentication types*
 - *Configure endpoints*
 - *Connect string options*
 - *Create via API*
 - *Example - single GET method*
 - *Example - rate limiting*
 - *Use*
 - *Using REST API Connections*
 - *Uses*
 - *Before you begin*
 - *Secure access*
 - *Reading data*
 - *Writing data*
 - *Reference*
-

The REST API connection type provides a generic interface to relational data available through REST APIs. Using this connection type, you can create connections to individual endpoints across hundreds of REST-based applications.

Limitations

- Import-only connection type
- A limited set of request methods is supported. See Method entry below.
- Using a passphrase when generating an SSH key is not supported.
- JSON response from API endpoint is required. API endpoints that return XML responses are not supported.
- OAuth 2.0 authentication is not supported.
- After the initial connection to an endpoint is made, the schema is cached. The schema is not updated again until the connection is edited.
- By default, the number of endpoints that you can specify to use an individual connection is 10. This limit can be modified.
 - For more information, see *Workspace Settings Page*.

Prerequisites

- You should identify the tables and (optional) data models for them that you wish to access.
- You should acquire the credentials to access your target endpoints for one of the supported authentication methods.
 - If you are using a key or token to access the endpoints, you should generate this token before you begin.
 - See below.

Enable connections on the same cluster

For security reasons, the Trifacta application prevents connections from services external to the product from within the same cluster by default. If you are creating connections to REST applications that are hosted on the same cluster as the Trifacta node, additional configuration is required. Please complete the following configuration steps.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameters and populate them as described below:

Parameter	Description
<code>data-service.allowedUrlReges</code>	<p>This parameter defines an array list of regular expressions for URLs that are permitted to connect to the data service. Please add the URL of the REST application to which you are connecting to the array list.</p> <p>The following example permits URLs that include <code>localhost</code> and <code>example</code>:</p> <pre>["localhost", "example"]</pre>
<code>data-service.forbiddenIPs</code>	<p>This parameter defines the list of IP addresses that are prevented from connecting to the data service. Please remove the following from this list:</p> <pre>10.0.0.0/8</pre>

3. Save your changes and restart the platform.

Configure

To create this connection, in the Connections page , select the Applications tab. Click the REST API card. See *Connections Page*.

Modify the following properties as needed:

Property	Description
Base URI	<p>The base URI for the endpoints to which you wish to connect through this connection. Example:</p> <pre>https://exampleserver.sharepoint.com/sites/SharePointTest</pre> <p>Tip: SSL access is supported over the HTTPS protocol.</p>
Connect String Options	<p>Apply any connection string options that are part of your authentication to REST API.</p> <p>A default string has been provided for you. For more information, see below.</p>
Authentication Type	The method by which you wish to authenticate to the endpoint. See "Authentication types" below.
API endpoints	Specify the endpoints to which to connect. For more information, see "Configure endpoints" below.
Test Connection	After you have defined the REST API credentials and connection string, you can validate those credentials.
Connection Name	Display name of the connection

Connection Description	Description of the connection, which appears in the application.
------------------------	--

Authentication types

The following types of authentication are supported for REST API connections. For each type, additional properties may require configuration.

Basic auth

A username/password combination is submitted as part of any request for authentication.

Property	Description
Username	Username to access the endpoints.
Password	Password associated with the username.

HTTP Header Based Auth

Authentication is submitted using a key/value pair submitted in the HTTP request header.

Property	Description
Header Key	Key for the header parameter used in authentication
Header Value	Credential associated with header authentication key

HTTP Query Based Auth

Authentication is submitted using a query parameter key/value pair submitted as part of the URL.

Property	Description
Query Key	Key for the query parameter used in authentication
Query Value	Credential associated with the query parameter authentication key

Configure endpoints

Each endpoint and method combination must be configured. To add an endpoint, click **Add endpoint**.

The properties are described below.

Property	Description
Method	<p>API request method to use. Supported methods:</p> <ul style="list-style-type: none"> • GET - read from the endpoint • POST - create a new instance of an object in the target application through the endpoint • PUT - modify an existing instance <div style="border: 1px solid red; padding: 10px; margin: 10px 0;"> <p>In some target systems, the PUT and POST methods are required for generating datasets for import. These methods should not be used for other uses cases, such as writing data back to the target system. REST API connections are supported for import only.</p> </div> <div style="border: 1px solid gray; padding: 10px; margin: 10px 0;"> <p>NOTE: Other methods are not supported for use.</p> </div>

URL Endpoint	<p>The endpoint that you are accessing using the specified method.</p> <div> Tip: The Base URI value and this value should form a complete URL. </div>
Table Name	(required) The name of the table with which you are interacting through this endpoint.
Data Model	<p>Select the type of model used for the selected table:</p> <ul style="list-style-type: none"> • Document - Data is returned as a document containing top-level elements which are represented as columns in the Trifacta application. Nested data is returned in aggregated form. • Relational - Data is returned in tabular form, in which each returned XPath represents an individual table containing a primary key and a foreign key linking to the parent document. • Flattened Documents - Data is stored as FlattenedArrays in the source system. A separate table is returned for each object array and is joined to its parent table. The parent table and each child table are joined into a single table for use in the Trifacta application . <p>For more information on these data model types, see https://cdn.cdata.com/help/DWE/jdbc/pg_RESTParsing.htm.</p>
Pagination	Select the type of pagination to request to the API endpoint. See "Pagination" below.
Advanced options: Custom Header	<p>(optional) You can insert a custom header in the request as a key/value pair.</p> <p>To add more headers, click Add.</p>
Advanced options: Query Parameter	<p>(optional) You can append a query parameter and value to the URL. These values are appended in the following form:</p> <div> <endpoint_url>?<key1>=<value1>&<key2>=<value2> </div> <p>To add more query parameters, click Add.</p>
Advanced options: XPath	(optional) You can specify an XPath to be queried of the URL.

Pagination

For the selected endpoint, you can specify the type of pagination in use by the target application. Specifying the pagination allows the Trifacta application to retrieve larger sets of records when pagination is in use. For more information on these pagination methods, see http://cdn.cdata.com/help/DWE/ado/pg_customschemaselect.htm.

None:

(default) No pagination is applied by the endpoint.

Next page URL:

When this method is selected, the URL of the next page of results is returned as part of the response body.

- **Page URL path** defines the XPath in the response to the attribute containing the URL of the next page.

Paging token:

A paging token may be returned as part of the response. To acquire the next page of results, this token must be submitted in the subsequent request as the value associated with the paging parameter.

- **Page token path** is the XPath to the token that must be submitted with the next request.
- **Page token param** is the parameter in the request into which the page token must be submitted.
- **More pages param** (optional) is used when the page token path must be submitted as a query parameter. This value defines the query parameter for which it is submitted.

Record offset:

Under this pagination method, subsequent pages of results can be queried based on defining the number of results (records) to offset with the query.

- **Page offset param** defines the query parameter where you can specify the page offset to query.
- **Page size param** defines the parameter in the request where you define the size in records of each request (page) of records.
- **Page size** defines the number of records to request in a page.

Page number:

Similar to record offset, this method queries results based on specified page numbers.

- **Page number param** defines the query parameter where you can specify the page number to query.
- **Page size param** defines the parameter in the request where you define the size in records of each page of records.
- **Page size** defines the number of records to request in a page.

Connect string options

Too many requests error

During execution, you may receive an error similar from the driver to the following:

```
PlatformErrors: Retries errors based on the exception message. E.g. Other=PlatformErrors="Too Many Requests"
MaximumRequestRetries: The number of times the driver will attempt to retry the request (Default 4)
```

In this case, the default wait time for retrying a request (2 seconds) is not enough time, and the requests are piling up. You can address this issue by inserting the following connect string options:

```
Other='RetryWaitTime=15000'
```

The above option sets the wait time before retrying to 15000ms (15 seconds). You can experiment with this value as needed.

For more information on available connect string options, see <https://cdn.cdata.com/help/DWE/ado/Connection.htm>.

Create via API

This connection can also be created using the API.

- Type: jdbc_rest
- Vendor: jdbc_rest

Example - single GET method

The following example request creates a REST API connection with the following characteristics:

- Query parameters are used for authentication
- A single endpoint is enabled:
 - Method: GET
 - Target: table1
 - dataModel: Document

```
{
  "vendor": "jdbc_rest",
  "vendorName": "jdbc_rest",
  "name": "REST API test",
```

```

"description": "",
"type": "jdbc",
"params": {
  "base_URI": "some base URI",
  "connectStrOpts": ""
},
"credentialType": "httpQueryBasedAuth",
"credentials": [
  {
    "queryKey": "user",
    "queryValue": "token"
  }
],
"endpoints": [
  {
    "tableName": "table1",
    "httpMethod": "get",
    "endpoint": "endpoint1",
    "requestBody": "",
    "xPath": "",
    "dataModel": "document",
    "headers": {},
    "queryParams": {}
  }
]
}

```

Example - rate limiting

The following example creates a REST API connection to polygon.io with the following characteristics:

- Query-based authentication using key/value pair
- Connect String Options:

```
Other='RetryWaitTime=15000';
```

- Wait for retry: 15000 milliseconds
- Single endpoint to GET stock ticker information:
 - DataModel: Document
 - XPath: \$.results
 - Rate limiting on the endpoint (maximum queries per minute): 1000
 - queryParams.date is a parameter that is passed in for this specific connection type.
 - Pagination: nextPageURL method

```

{
  "vendor": "jdbc_rest",
  "vendorName": "jdbc_rest",
  "name": "REST API",
  "description": "",
  "type": "jdbc",
  "params": {
    "base_URI": "https://api.polygon.io/",
    "connectStrOpts": "Other='RetryWaitTime=15000';"
  },
  "credentialType": "httpQueryBasedAuth",
  "credentials": [
    {
      "queryKey": "apiKey",
      "queryValue": "someKey"
    }
  ],
  "endpoints": [
    {

```

```

    "tableName": "tickers",
    "httpMethod": "get",
    "endpoint": "/v3/reference/tickers",
    "requestBody": "",
    "xPath": "$./results",
    "dataModel": "document",
    "headers": {},
    "queryParams": {
      "limit": "1000",
      "date": "2021-11-04T00:00:00Z"
    },
    "pagination": {
      "pageurlpath": "$./next_url",
      "paginationType": "nextPageURL"
    }
  }
]
}

```

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Use

You can import datasets from REST API through the Import Data page. See *Import Data Page*.

Tip: You can perform joins and unions using custom SQL as part of your initial request for data. It may be easier to import the tables as separate datasets and then to perform the join or union within the Trifacta application.

Using REST API Connections

This section describes how you interact through Trifacta® with your REST data.

Uses

Trifacta can use REST API connections for the following tasks:

1. Import datasets

Before you begin

- **Read Access:** You must have credentials to create access the specific endpoints required to retrieve your data.
- **Write Access:** Not supported

Secure access

SSL is available over HTTPS for REST API connections.

Reading data

You can create a Trifacta dataset from the following data models:

1. documents
2. flattened documents
3. relational tables

These source objects are represented as tables in the import browser and as grid data in the Transformer page.

For more information, see *Database Browser*.

Writing data

Not supported.

NOTE: Do not use the `PUT` and `POST` methods to write data back into the target system.

Reference

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Not supported	Not supported	Not supported

Enable Teradata Access

Contents:

- *Limitations*
 - *Create Teradata Connection*
 - *Troubleshooting*
 - *Testing*
-

This section provides information on how to enable connection to Teradata .

- Teradata provides Datawarehousing & Analytics solutions and Marketing applications. The Teradata supports all of their Data warehousing solutions. For more information, see <http://www.teradata.com>.
- For more information on supported versions, see *Connection Types*.

This connection supports reading and writing. You can create multiple Teradata connections in the Trifacta application.

Supported Versions: 14.10+

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Supported	Supported	Supported

Limitations

- By default, Teradata does not permit the publication of datasets containing duplicate rows. Workarounds:
 - Your final statement for any recipe that generates results for Teradata should include a `Remove duplicate rows` transformation.

NOTE: The above transformation removes exact, case-sensitive duplicate rows. Teradata may still prevent publication for case-insensitive duplicates.

- It's possible to change the default writing method to Teradata to enable duplicate rows. For more information, contact *Alteryx Support*.
- When creating custom datasets using SQL from Teradata sources, the `ORDER BY` clause in standard SQL does not work. This is a known issue.

Create Teradata Connection

For more information on creating a Teradata connection, see *Teradata Connections*.

Troubleshooting

Error	Description
Duplicate row error	<p>This error occurs when duplicate rows are being inserted during publishing to Teradata.</p> <div><p>Workaround: All inserted rows must be unique, or the Teradata tables must be <code>MULTISET</code>. To configure Trifacta to use <code>MULTISET</code> tables, please contact <i>Alteryx Support</i>.</p></div>

Testing

Steps:

1. After you create your connection, load a small dataset based on a table in the connected Teradata. See *Import Data Page*.
2. Perform a few simple transformations to the data. Run the job. See *Transformer Page*.
3. Verify the results.

For more information, see *Verify Operations*.

Teradata Connections

Contents:

- *Limitations*
 - *Prerequisites*
 - *Create Teradata Connection*
 - *Connection URL*
 - *Driver Information*
 - *Create via API*
 - *Using Teradata Connections*
 - *Uses of Teradata*
 - *Before you begin using Teradata*
 - *Writing to Teradata*
 - *SQL Syntax*
-

You can create connections to your Teradata instance from Trifacta®.

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Limitations

- Tables inside the DBC database are not listed due to technical constraints; however you can access the tables through custom SQL.
- Custom SQL is supported. The Custom SQL should be in the following format:

```
SELECT * FROM <DATABASE_NAME>.<Table_NAME>;
```

- When using custom SQL, the keyword `LIMIT` is not supported by Teradata. Use the keyword `TOP` to get the required number of rows similar to the following:

```
SELECT TOP <NUMBER_OF_ROWS> * FROM <DATABASE_NAME>.<TABLE_NAME>;
```

- In the `Connect String Options`, the parameters must be separated by commas:

```
Database=DBC,DBC_PORT=1025
```

Prerequisites

- Additional setup is required. For more information, see *Enable Teradata Access*.

Create Teradata Connection

To create this connection, in the Connections page, select the Databases tab. Click the Teradata card. See *Connections Page*.

Modify the following properties as needed:

Property	Description
Host	Enter the host name of Teradata. Example value: <div>buick1.teradata.ws</div>
Port	Set this value to the port number through which to access Teradata. By default, this values is set to 1025.
Connect String Options	(Optional) You can specify additional options used to connect as a string value. Connect string options are submitted in the following format: <div>option1=value1,option2=value2</div>
Enable Data Encryption	When enabled, the data exchanged between the Teradata JDBC driver and the database is encrypted.
User Name	The username used to connect.
Password	The password associated with the username.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can verify that the Trifacta application can use them to connect to the database.
Default Column Data Type Inference	Set to disabled to prevent the product from applying its own type inference to each column on import. The default value is enabled.
Connection Name	Display name of the connection.
Connection Description	(Optional) Description of the connection, which appears in the application.

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:teradata://<host>/DBS_PORT=<port>,<connect-string-options>
```

where:

Parameter	Description
<host>	Host URL of the database server.
<port>	Port number for the database server. Typically, this value is 1025 for Teradata.
<connect-string-options>	Connect string options, which are submitted in the following format: <div>option1=value1,option2=value2</div>

The Connection URL is mostly built up automatically using cluster configuration for the platform.

Use Data Encryption

When Data Encryption is enabled for the connection, the following is automatically appended to the connect string options:

```
,ENCRYPTDATA=ON
```

Driver Information

This connection uses the following driver:

- **Driver name:** `com.teradata.jdbc.TeraDriver`
- **Driver version:** `com.teradata:terajdbc4:17.00.00.03`
- **Driver documentation:** <https://developer.teradata.com/connectivity/reference/jdbc-driver>

Create via API

This connection can also be created using the API.

```
"vendor": "teradata",  
"vendorName": "Teradata",  
"type": "jdbc"
```

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Using Teradata Connections

Uses of Teradata

Trifacta can use for the following tasks:

- Create datasets by reading from tables.
- Writing back to Teradata.

Before you begin using Teradata

Read Access:

Your administrator must configure read permissions.

Writing to Teradata

Supported.

SQL Syntax

The following syntax requirements apply to this connection.

Object delimiter: double-quote or no delimiter

Example syntax:

Double quotes can be used around database names, table names, or column names.

Note that references to specific values must be single-quoted for columns of String data type.

```
SELECT "column1","column2" FROM "databaseName"."tableName" WHERE "column3" = 'my_value';
```

For more information on SQL in general, see *Supported SQL Syntax*.

MongoDB Connections

Contents:

- *Prerequisites*
 - *Limitations*
 - *Create Connection*
 - *MongoDB*
 - *MongoDB Atlas*
 - *Create connection via API*
 - *Connect string options*
 - *Using MongoDB*
 - *MongoDB Data Organization Hierarchy*
 - *Database Uses*
 - *Read Data*
 - *Data Type Mappings*
 - *Access/Read*
 - *Write/Publish*
-

You can create connections to MongoDB and MongoDB Atlas connections through Trifacta application . These connections enable to read data from the MongoDB workspace.**Supported Versions:** n/a

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Not supported	Not supported	Not supported

Prerequisites

- MongoDB supports basic (username/password) authentication.

Limitations

NOTE: During normal selection or import of an entire table, you may encounter an error indicating a problem with a specific column. Since some tables require filtering based on a particular column, data from them can only be ingested using custom SQL statements. In this case, the problematic column can be used as a filter in the WHERE clause of a custom SQL statement to ingest the table.

- For more information, please consult the CData driver documentation for the specific table.
- For more information on using custom SQL, see *Create Dataset with SQL*.

NOTE: For filtering date columns, this connection type supports a set of literal functions on dates. You can use these to reduce the volume of data extracted from the database using a custom SQL query. For more information, see the `pg_dateliteralfunctions.htm` page in the driver documentation for this connection type.

- This connection is read-only.

Create Connection

MongoDB

To create a MongoDB connection, please specify the following properties:

Property	Description
Host	Name of the host.
Port	Set this value to the port number through which to access MongoDB. By default, this value is 27017.
Database	The database that you want to read
Auth Database	Name of the MongoDB database used for authentication
Replica Set	(Optional) Comma-separated list of secondary servers in the replica set, specified by address and port. A replica set is a group of mongoDB processes that maintain the same data set. Replica sets provide redundancy and high availability and are the basis for all production deployments. For more information, see https://docs.mongodb.com/manual/replication/ .
Secondary Reads	Enable this checkbox if you want to read from secondary (slave) servers.
Use SSL	Enable this checkbox if you want to connect using SSL.
Connect String Options	(Optional) You can specify additional options used to connect as a string value. The following option sets the connection timeout in milliseconds: <div>Timeout=0;</div> The default value is 0, which disables connection timeouts. See below for more information.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can verify that the Trifacta application can use them to connect to the database.
Default Column Data Type Inference	Set to disabled to prevent the product from applying its own type inference to each column on import. The default value is enabled .
Connection Name	Display name of the connection
Connection Description	(Optional) Description of the connection, which appears in the application.

MongoDB Atlas

To create a MongoDB Atlas connection, please specify the following properties:

Property	Description
Host	Name of the host.
Port	Set this value to the port number through which to access MongoDB. By default, this value is 27017.
Database	The database that you want to read
Replica Set	(Optional) Comma-separated list of secondary servers in the replica set, specified by address and port. A replica set is a group of mongoDB processes that maintain the same data set. Replica sets provide redundancy and high availability and are the basis for all production deployments. For more information, see https://docs.mongodb.com/manual/replication/ .
Secondary Reads	Enable this checkbox if you want to read from secondary (slave) servers.

Connect String Options	<p>(Optional) The option sets the connection timeout in milliseconds:</p> <pre>Timeout=0;</pre> <p>The default value is 0, which disables connection timeouts. See below for more information.</p>
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can verify that the Trifacta application can use them to connect to the database.
Default Column Data Type Inference	Set to <code>disabled</code> to prevent the product from applying its own type inference to each column on import. The default value is <code>enabled</code> .
Connection Name	Display name of the connection
Connection Description	(Optional) Description of the connection, which appears in the application.

For more information on these settings, see <http://cdn.cdata.com/help/DGG/jdbc/default.htm>.

Create connection via API

Depending on your product edition, you can create connections of this type.

MongoDB:

```
"vendor": "mongodb",
"vendorName": "MongoDB",
"type": "jdbc"
```

MongoDB Atlas:

```
"vendor": "mongodb_atlas",
"vendorName": "MongoDB Atlas",
"type": "jdbc"
```

<https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Connect string options

Connection timeout

By default, the supported driver applies a connection timeout to MongoDB of 0 seconds. As needed, you can modify the connection timeout through connect string options:

```
Timeout=<value_in_seconds>;
```

where:

`<value_in_seconds>` corresponds to the number of seconds for the time.

Flattening Documents

Documents can contain other documents, which enables the storage of nested data. You can control the flattening of nested objects and arrays through the CData driver through Connect String Options.

NOTE: Columns that have been flattened can be accessed or referenced using custom SQL queries. Additional information is below.

Flatten Objects:

By default, the CData driver flattens nested Objects. As needed, you can set FlattenObjects to `false` to disable this behavior.

For more information, see http://cdn.cdata.com/help/DGG/jdbc/RSBMongodb_p_FlattenObjects.htm.

Flatten Arrays:

By default, CData driver does not flatten Arrays.

- As needed, you can configure the number of elements that you want to have returned in your flattened arrays.
- To flatten all elements of all arrays, set FlattenArrays to `-1`.

For more information, see http://cdn.cdata.com/help/DGG/jdbc/RSBMongodb_p_FlattenArrays.htm.

Referencing flattened columns:

If you have flattened Objects or Arrays, you can reference these columns using square brackets in your custom SQL queries.

Example of flattened Object:

```
SELECT [address.city] FROM my_table;
```

Example of flattened Array:

```
SELECT * FROM my_table WHERE [hobbies.0]='cricket';
```

Driver Information

For more information on CData JDBC drivers, see <http://cdn.cdata.com/help/DGG/jdbc/default.htm>.

Using MongoDB

MongoDB is a NoSQL document database that provides high performance, availability, and scalability.

MongoDB Data Organization Hierarchy

MongoDb has a two-level data hierarchy:

```
+ Schema1
  + Collection1
  + Collection2
+ Schema2
  + Collection3
  + Collection4
```

- **Schema** roughly corresponds to a database.

- **Collection** roughly corresponds to a table.
 - A collection is composed of documents. A **Document** is a binary JSON representation of the fields and values of a row.

Database Uses

For more information on interacting with databases, see *Using Databases*.

Read Data

You can import datasets from MongoDB through the Import Data page. See *Import Data Page*.

Data Type Mappings

NOTE: The Trifacta® data types listed in this section reflect the raw data type of the converted column. Depending on the contents of the column, the Transformer Page may re-infer a different data type, when a dataset using this type of source is loaded.

Access/Read

When data is imported from MongoDB, the supported data types from the source are converted to corresponding data types supported by the application. For more information, see *Type Conversions*.

Source Data Type	Supported	Trifacta data type
ObjectId	Y	String
Regex	Y	String
String	Y	String
Binary	Y	String
Integer	Y	Integer
Timestamp	Y	Datetime
Double	Y	Float
Array	Y	String
Bool	Y	bool
Null	Y	String
Date	Y	Datetime

Write/Publish

Not supported.

Tableau Server Connections

Contents:

- *Limitations*
- *Enable Hyper format*
- *Configure Permissions*
- *Create Tableau Server Connection*
 - *Create through application*
 - *Create through APIs*

This section describes the basics of creating Tableau Server connections from within the application.

NOTE: You can export Tableau Server files as part of exporting results from the platform. For more information, see *Publishing Dialog*.

Supported Versions: 10.5.x and later

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Not supported	Not supported	Not supported
Write	Supported	Supported	Supported

Limitations

- This connection type only enables publication.
 - You cannot read data from Tableau Server.
 - When created in the application, publish-only connections must be created through the Connections page.

Enable Hyper format

Hyper format generation is enabled by default. To enable the generation of results into Hyper format, please verify the following:

Steps:

1. Login as an administrator.
2. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
3. Locate the following setting:

Hyper output format

4. Set it to `Enabled`.
5. No other configuration is required.

Configure Permissions

The user who is publishing to Tableau Server must have exec permissions on the temporary directory on the backend datastore. This directory is used to write the intermediate file format locally, before it is published to Tableau Server. For more information, see *Supported File Formats*.

Create Tableau Server Connection

Create through application

Any user can create a Tableau Server connection through the application.

NOTE: Only an administrator can make a Tableau Server connection available for all users.

Steps:

1. In the left nav bar, select the Connections icon. See *Connections Page*.
2. In the Connections page, click **Create Connection**. See *Create Connection Window*.
3. In the Create Connection window, click the **Tableau Server** connection card.
4. Specify the properties for your Tableau Server.

Property	Description
Server URL	<p>The URL to the Tableau Server to which you are connecting. To specify an SSL connection, use <code>https://</code> for the protocol identifier.</p> <div><p>NOTE: By default, this connection assumes that the port number is 80. To use a different port, you must specify it as part of the Server name value: <code>http://<Tableau_Server_URL>:<port_number></code></p></div>
Site	<p>Enter the value that appears after <code>/site/</code> in your target location.</p> <p>Example target URL:</p> <div><code>https://tableau.example.com/#/site/MyNewTargetSite</code></div> <p>Enter the following for the Site setting:</p> <div><code>MyNewTargetSite</code></div>
User Name	The username to use to connect.
Password	The password associated with the username.
Test Connection	Click this button to test the connection that you have specified.
Connection Name	The name of the connection as you want it to appear in the user interface.
Description	This description is displayed in the user interface.

For more information, see *Create Connection Window*.

5. Click **Save**.

Create through APIs

You can create this connection type through the APIs:

API: *API Reference*

- Type: jdbc
- Vendor: tableau

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Salesforce Connections

Contents:

- *Limitations*
 - *Prerequisites*
 - *Enable*
 - *Configure*
 - *Connect to Sandbox account*
 - *Connect string options*
 - *Create via API*
 - *Use*
 - *Using Salesforce Connections*
 - *Uses of Salesforce*
 - *Before you begin using Salesforce*
 - *Secure access*
 - *Storing data in Salesforce*
 - *Reading from Salesforce*
 - *Writing to Salesforce*
 - *Reference*
-

You can create connections to your Salesforce instance from Trifacta®. This connector is designed as a wrapper around the Salesforce REST API.

Limitations

NOTE: During normal selection or import of an entire table, you may encounter an error indicating a problem with a specific column. Since some tables require filtering based on a particular column, data from them can only be ingested using custom SQL statements. In this case, the problematic column can be used as a filter in the WHERE clause of a custom SQL statement to ingest the table.

- For more information, please consult the CData driver documentation for the specific table.
- For more information on using custom SQL, see *Create Dataset with SQL*.

NOTE: For filtering date columns, this connection type supports a set of literal functions on dates. You can use these to reduce the volume of data extracted from the database using a custom SQL query. For more information, see the `pg_dateliteralfunctions.htm` page in the driver documentation for this connection type.

- This is a read-only connection.
- Single Sign-On (SSO) is not supported.
- Custom domains are not supported.
- You cannot ingest Salesforce tables that require mandatory filters.

Prerequisites

- The account used to login from Trifacta must access Salesforce through a security token.

NOTE: Please contact your Salesforce administrator for the Server Name and the Security Token values.

- The logged-in user must have required access to the tables and schema.
- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Enable

- General relational connectivity must be enabled. For more information, see *Relational Access*.
- This connection type utilizes OAuth 2.0 for authentication.

NOTE: OAuth 2.0 authentication requires additional configuration specific to the connection type.

For more information, see *Enable OAuth 2.0 Authentication*.

Configure

To create this connection, in the Connections page, select the Applications tab. Click the Salesforce card. See *Connections Page*.

Modify the following properties as needed:

Property	Description
Server Name	Enter the host name of your Salesforce implementation. Example value: <div>exampleserver.salesforce.com</div>
Connect String Options	Apply any connection string options that are part of your authentication to Salesforce. For more information, see below.
Credential Type	Select the type of credentials to provide with the connection: <ul style="list-style-type: none"> • SecurityToken - apply the security token that has been generated within the account to authenticate to Salesforce. • OAuth 2.0 - use OAuth 2.0 client connect to Salesforce. An OAuth 2.0 client may already be defined in the Trifacta application. <p>NOTE: After you have specified the connection to use OAuth 2.0, click Authenticate to validate the connection with the target datastore. If you have modified the connection, click Re-authenticate to validate the new connection definition. You must re-authenticate if you receive an expired tokens message.</p>
OAuth 2.0 Client	(OAuth 2.0 credential type) Select the OAuth 2.0 client to use.
User Name	(SecurityToken credential type) Username to use to connect to the database.
Password	(SecurityToken credential type) Password associated with the above username.
Security Token generated in account	(SecurityToken credential type) Paste the security token associated with the account to use for this connection.
Test Connection	(SecurityToken credential type) After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Default Column	Set to disabled to prevent the platform from applying its own type inference to each column on import. The default value is enabled .

Data Type Inference	
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Connect to Sandbox account

If you are connecting to a Salesforce sandbox account, the following property values must be modified:

Property	Sandbox Value
Server Name	<div>test.salesforce.com</div>
Connect String Options	<p>Please append the following to your connect string options:</p> <div>UseSandbox=true;</div> <p>For more information, see below.</p>
OAuth 2.0 client	Select Salesforce Sandbox.
User Name	<p>(Security token credential type) Append the name of the sandbox environment to the end of the username. In the following, the sandbox environment is demo:</p> <div>exampleUser@example.com.demo</div>

For OAuth 2.0 authentication:

The OAuth 2.0 client must be modified to use the following values from test.salesforce.com:

Property	Setting
Authorization URL	<div>"https://test.salesforce.com/services/oauth2/authorize"</div>
Token Url	<div>"https://test.salesforce.com/services/oauth2/token"</div>

Connect string options

Connection timeout

By default, the supported driver applies a connection timeout to Salesforce of 60 seconds. As needed, you can modify the connection timeout through connect string options:

```
timeout=<value_in_seconds>
```

where:

<value_in_seconds> corresponds to the number of seconds for the time.

NOTE: Although it is not recommended, you can set this value to 0 to disable timeouts.

Schema caching

By default, the connection driver uses schema caching to speed up ingestion. To surface changes to Salesforce tables/schema immediately, you can use the following options to disable schema caching by the connection:

```
Other='cachemetadatatable=false;cachemetadatatablecolumns=false;'
```

Create via API

This connection can also be created using the API.

NOTE: If you are using OAuth 2.0 authentication for this type, you cannot create connections via API.

- Type: jdbc
- Vendor: salesforce

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Use

You can import datasets from Salesforce. See *Database Browser*.

Using Salesforce Connections

Uses of Salesforce

Trifacta can use Salesforce for the following tasks:

1. Create datasets by reading from Salesforce tables.

Before you begin using Salesforce

Read Access:

- Your Salesforce administrator must configure read permissions.
- You must acquire a Salesforce security token for use with the Salesforce connection.

Secure access

SSL is the default connection method.

Storing data in Salesforce

Your Salesforce administrator should provide database access for storing datasets. Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: Trifacta does not modify source data in Salesforce. Datasets sourced from Salesforce are read without modification from their source locations.

Reading from Salesforce

When Trifacta connects to your Salesforce instance, the application can read from all Salesforce objects that are accessible through the Salesforce account in use, including:

- Salesforce objects and fields are mapped to tables and columns
- Standard and custom objects

NOTE: The names of custom objects are appended with the value `_c`.

- Audit columns
- System fields

NOTE: Unquoted identifiers are converted to uppercase during import.

You can create a Trifacta dataset from a table stored in Salesforce.

Writing to Salesforce

Not supported.

Reference

Supported Versions: n/a

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Not supported	Not supported	Not supported

SharePoint Connections

Contents:

- *Limitations*
 - *Prerequisites*
 - *Enable*
 - *Configure*
 - *Connect string options*
 - *Create via API*
 - *Use*
 - *Using SharePoint Connections*
 - *Uses of SharePoint*
 - *Before You Begin Using SharePoint*
 - *Secure access*
 - *Storing data in SharePoint*
 - *Reading from SharePoint*
 - *Writing to SharePoint*
 - *Reference*
-

You can create connections to your Microsoft SharePoint instance from Trifacta®.

You can create connections to:

- SharePoint On-Premises installations in your enterprise infrastructure
- SharePoint Online

NOTE: This connection supports reading from and writing to SharePoint lists.

For more information on Microsoft SharePoint, see <https://www.microsoft.com/en-us/microsoft-365/sharepoint/collaboration>.

Limitations

NOTE: During normal selection or import of an entire table, you may encounter an error indicating a problem with a specific column. Since some tables require filtering based on a particular column, data from them can only be ingested using custom SQL statements. In this case, the problematic column can be used as a filter in the WHERE clause of a custom SQL statement to ingest the table.

- For more information, please consult the CData driver documentation for the specific table.
- For more information on using custom SQL, see *Create Dataset with SQL*.

NOTE: For filtering date columns, this connection type supports a set of literal functions on dates. You can use these to reduce the volume of data extracted from the database using a custom SQL query. For more information, see the `pg_dateliteralfunctions.htm` page in the driver documentation for this connection type.

- Single Sign-On (SSO) is not supported.

- Column names are not validated on publishing.
- The SharePoint connection uses SharePoint APIs. As a result, transaction management and rollbacks are not supported.
- No schema validation is performed as part of writing results to SharePoint Lists.

Prerequisites

- The logged-in user must have required access to the tables and schema.
- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Enable

- General relational connectivity must be enabled. For more information, see *Relational Access*.

Configure

To create this connection, in the Connections page, select the Applications tab. Click the **SharePoint** card. See *Connections Page*.

Modify the following properties as needed:

Property	Description
SharePoint URL	Enter the URL for your SharePoint Site or sub-site. Example value: <div>https://exampleserver.sharepoint.com/sites/SharePointTest</div>
SharePoint Edition	Product edition of SharePoint in use: <ul style="list-style-type: none"> • <code>SharePointOnPremise</code> - Use this option if you have an on-premises installation of SharePoint to which you can connect within your enterprise infrastructure. • <code>SharePointOnline</code> - Use this option if you are connecting to SharePoint Online.
Auth Scheme	Authentication scheme: <ul style="list-style-type: none"> • <code>Basic</code> - (for SharePointOnline) username and password • <code>NTLM</code> - (for SharePointOnPremise) Windows-based authentication scheme for on-premises deployments.
User Name	Username to use to connect to SharePoint.
Password	Password associated with the above username.
Test Connection	After you have defined the SharePoint Edition, credentials, and connection string, you can validate those credentials.
Additional Connect String Options	Apply any connection string options that are part of your authentication to SharePoint. A default string has been provided for you. For more information, see below.
Default Column Data Type Inference	Set to <code>disabled</code> to prevent the platform from applying its own type inference to each column on import. The default value is <code>enabled</code> .
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Connect string options

The following connection string is provided for you:

```
AutoCache=false;CacheMetadata=false;CacheTolerance=1;timeout=0;ShowPredefinedColumns=false
```

Parameter	Description
AutoCache	When enabled, the connection leverages any data that is automatically cached for each table. The default is <code>false</code> .
CacheMetadata	When enabled, table metadata can be retrieved from the SharePoint cache. The default is <code>false</code> .
CacheTolerance	This setting defines the duration in hours that objects are permitted to live in the cache. The default is <code>1</code> .
timeout	<p>This setting defines the number of seconds that a query to the SharePoint database is allowed to run without a response. The SharePoint default timeout is <code>60</code>, which may cause complex queries of larger datasets to timeout.</p> <p>The default value in the Connect String Options is <code>0</code>, which means that there is no enforced timeout. Other timeouts may apply.</p>
ShowPredefinedColumns	When enabled, users of the connection are permitted to view the columns that are created with the table, such as Created By and Modified By columns. The default is <code>false</code> .

For more information, see <http://cdn.cdata.com/help/RSG/jdbc/Connection.htm>.

Create via API

This connection can also be created using the API.

- Type: `jdbc`
- Vendor: `sharepoint`

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Use

You can import datasets from SharePoint. See *Database Browser*.

Using SharePoint Connections

This section describes how you interact through Trifacta® with your SharePoint Lists.

- SharePoint is a content management system for sharing and collaborating across the enterprise. For more information, see <https://sharepoint.microsoft.com>.
- In SharePoint, data is stored in an object called a List. Lists can also include non-tabular data.

NOTE: Non-tabular data and some SharePoint List columns are converted to strings on import. For more information, see *SharePoint Data Type Conversions*.

Uses of SharePoint

Trifacta can use SharePoint for the following tasks:

1. Import datasets by reading from SharePoint Lists.

2. Write to SharePoint Lists with your job results.

Before You Begin Using SharePoint

- **Read Access:** Your SharePoint administrator must configure read permissions.
- **Write Access:** You can write and publish jobs results to SharePoint.

Secure access

SSL is available over HTTPS for SharePoint connections.

Storing data in SharePoint

Your SharePoint administrator should provide database access for storing datasets. Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: Trifacta does not modify source data in SharePoint. Datasets sourced from SharePoint are read without modification from their source locations.

Reading from SharePoint

You can create a Trifacta dataset from a List stored in SharePoint.

NOTE: Reading data is supported for SharePoint Lists only.

For more information, see *Database Browser*.

Writing to SharePoint

You can write back data to SharePoint using one of the following methods:

- Job results can be written directly to SharePoint as part of the normal job execution. Create a new publishing action to write to SharePoint. See *Run Job Page*.
- For more information on how data is converted to SharePoint, see *SharePoint Data Type Conversions*.

Data Validation issues:

NOTE: Some Trifacta data types do not map exactly to SharePoint List data types. These differences may appear when writing to a new SharePoint List. For more information, see *SharePoint Data Type Conversions*.

NOTE: Column name validation is not supported.

- No validation is performed for the connection and any required permissions during job execution. So, you can be permitted to launch your job even if you do not have sufficient connectivity or permissions to access the data. The corresponding publish job fails at runtime.
- Prior to publication, no validation is performed on whether a target is a table or a view, so the job that was launched fails at runtime.

Reference

Supported versions: n/a

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Google Analytics Connections

Contents:

- *Limitations and Requirements*
 - *Create Connection*
 - *via Trifacta application*
 - *Connect String Options*
 - *Data Type Conversions*
-

Google Analytics is a web analytics service offered by Google that tracks and reports website traffic. For more information, see <https://analytics.google.com/analytics/web/provision/#/provision>.

Limitations and Requirements

NOTE: During normal selection or import of an entire table, you may encounter an error indicating a problem with a specific column. Since some tables require filtering based on a particular column, data from them can only be ingested using custom SQL statements. In this case, the problematic column can be used as a filter in the WHERE clause of a custom SQL statement to ingest the table.

- For more information, please consult the CData driver documentation for the specific table.
- For more information on using custom SQL, see *Create Dataset with SQL*.

NOTE: For filtering date columns, this connection type supports a set of literal functions on dates. You can use these to reduce the volume of data extracted from the database using a custom SQL query. For more information, see the `pg_dateliteralfunctions.htm` page in the driver documentation for this connection type.

NOTE: Most interactions with the Google Analytics datastore are formed as custom SQL queries. More information on syntax and examples is provided below.

- OAuth 2.0 authentication is required.
 - An OAuth 2.0 web client must be available for use in the Trifacta application. For more information, see *OAuth 2.0 for Google Analytics*.
 - You cannot create OAuth 2.0 connections via API.
- The following schema is supported: `UniversalAnalytics`.

Tip: This schema was previously called `GoogleAnalytics` by the driver vendor.

- Google Analytics allows up to 10 metrics and seven dimensions in a single query.
 - When issuing a query that selects all columns, only the default Metric columns are selected for tables with more than 10 Metrics.
 - The default Dimensions are used unless you explicitly select other dimension columns.
- All reports in Google Analytics must cover a specific date range.
 - The default behavior is to pull the last month of data if the `StartDate` and `EndDate` inputs are left unset.
 - To override this behavior, the values can be set directly in the query.

Create Connection

via Trifacta application

When you create the connection, please review the following properties and specify them accordingly:

Connection Property	Description
View Id	<p>Unique identifier of the view</p> <div>Tip: To acquire your View Id, login to Google Analytics and navigate to your site data. Click the name of your site in the top menu bar. The identifiers for the available views are listed as numeric values in the right column.</div>
Connect String Options	<p>The following is the default connect string option:</p> <div><code>Timeout=0;SupportEnhancedSQL=true;</code></div> <ul style="list-style-type: none">• The first option sets the connection timeout in seconds. Setting this value to 0 disables timeouts.• For more information on the second option, see below.
OAuth2 Client	<p>The client is displayed.</p> <div>NOTE: When you create the connection in this window, you must click Authenticate, which authenticates to the app. This step is required.</div>
Default Column Data Type Inference	<p>Leave this value as <code>Enabled</code>.</p>

For more information, see the driver documentation <http://cdn.cdata.com/help/DAG/jdbc/default.htm>.

Connect String Options

Enhanced SQL

By default, the following connect string options is included. This option enables an enhanced form of SQL support for the connection.

```
SupportEnhancedSQL=true;
```

When this feature is enabled:

- In-memory processing that is handled by default in the querying application (Trifacta application) is passed to Google Analytics for processing.
- This feature enables the use of advanced SQL expressions, such as the use of more predicates, joins, and aggregations.

For more information on enhanced SQL for Google Analytics, see https://cdn.cdata.com/help/DAG/jdbc/RSBGoogleAnalytics_p_SupportEnhancedSQL.htm.

Data Type Conversions

For more information, see the driver documentation <http://cdn.cdata.com/help/DAG/jdbc/>.

Example queries:

To import data from Google Analytics, you typically create a custom SQL SELECT statement. Example:

```
SELECT distinct Date, Sessions, NewUsers, BounceRate, PageviewsPerSession, AvgSessionDuration, Browser, FROM  
GoogleAnalytics.Traffic where StartDate = '90DaysAgo' and EndDate ='Today'
```

NOTE: The `distinct` keyword is currently required in default configuration when referencing specific values in a `where` clause. Optionally, you can enable the use of enhanced SQL for this connection, which eliminates the need for the `distinct` keyword. See "Connect String Options" above.

- Do not apply quotes, double-quotes, or brackets around field or table values.
- Literal values should be in single quotes.
- `StartDate` and `EndDate` parameters can be used to specify a date range.
 - You can also use date literal functions to specify date ranges. For more information, see http://cdn.cdata.com/help/DAG/jdbc/pg_dateliteralfunctions.htm.

For more information:

- SQL syntax: http://cdn.cdata.com/help/DAG/jdbc/pg_overview.htm
- SQL examples: http://cdn.cdata.com/help/DAG/jdbc/pg_retrievingdata.htm

DB2 Connections

Contents:

- *Prerequisites*
 - *Configure*
 - *Create connection via API*
 - *Reference*
 - *Connection URL*
 - *Driver Information*
 - *Use*
 - *Data Conversion*
-

You can create connections to one or more DB2 databases from Trifacta®.

NOTE: This method of creating DB2 connections is supported for customer-managed installations of Trifacta.

NOTE: Only connections to DB2 for Windows and Unix/Linux are supported.

Supported Versions: v10.5.5

Supported Environments:

Operation	Trifacta	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Not supported	Not supported	Not supported

Prerequisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the DB2 card.
- You can also create connections through the Connections page. See *Connections Page*.

Modify the following properties as needed:

Property	Description
Host	Enter your hostname. Example:

	<div>myDB2.example.com</div>
Port	Set this value to 50000.
Connect String Options	Please insert any connection options as a string here.
Database Name	Enter the name of the DB2 database to which to connect.
User Name	(basic credential type only) Username to use to connect to the database.
Password	(basic credential type only) Password associated with the above username.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Default Column Data Type Inference	Set to disabled to prevent the product from applying its own type inference to each column on import. The default value is enabled.
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Create connection via API

This connection can also be created using the API.

API: *API Reference*

- Type: jdbc
- Vendor: db2

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnection>

Reference

Connection URL

The properties that you provide are inserted into the following URL, which connects Trifacta to the connection:

```
jdbc:trifacta:db2://<host>:<port>
```

Connect string options

The connect string options are optional. If you are passing additional properties and values to complete the connection, the connect string options must be structured in the following manner:

```
;<prop1>=<val1>;<prop2>=<val2>;...
```

where:

- <prop> : the name of the property
- <val> : the value for the property

delimiters:

- ; : any set of connect string options must begin and end with a semi-colon.
- = : property names and values must be separated with an equal sign (=).

Driver Information

This connection uses the following driver:

- **Driver name:** `com.trifacta.connect.jdbc.db2.DB2Driver`

NOTE: The driver in use is a proprietary version of the driver listed in the documentation. The behavior and property name are the same.

- **Driver version:** `DataDirect 5.1.4`
- **Driver documentation:**
<https://docs.progress.com/bundle/datadirect-connect-jdbc-51/page/DB2-Driver.html>

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *DB2 Data Type Conversions*.

Configure Connectivity

Contents:

- *Enable*
 - *Data Service*
 - *Relational Features*
 - *Custom SQL Query*
 - *JDBC Ingestion*
 - *Append hadoop principal to logged queries*
 - *Enable Driver Logging*
 - *Configure Security*
 - *Enable SSO Connections*
 - *Type Inference*
 - *Enable OAuth 2.0 Connectivity*
-

This section covers the following areas around general connectivity of Trifacta®.

Additional configuration may be required for individual connection types. For more information, see *Connection Types*.

Enable

The platform automatically enables connectivity to relational databases for reading in datasets and writing results back out.

NOTE: Relational connectivity requires the use of an encryption key file, which must be created and deployed before you create relational connections. For more information, see *Create Encryption Key File* in the Install Guide.

Data Service

The platform streams records from relational sources through the data service. These records are applied to transformation and sampling jobs on the Photon running environment, which is native to the Trifacta node.

Tip: In general, you should not have to modify settings for the data service. However, if you are experiencing general performance issues or issues with specific connection types, you may experiment with settings in the data service.

For more information, see *Configure Data Service* in the Configuration Guide.

Relational Features

Custom SQL Query

To enhance performance of your relational datasets, you can enable the use of custom SQL queries against your relational datasources, which allows you to pre-filter your datasets before you ingest them into the platform. This feature is enabled by default, but additional configuration can be applied. See *Enable Custom SQL Query*.

JDBC Ingestion

By default, the platform ingests data from your relational datasources to the base storage layer for faster job execution. See *Configure JDBC Ingestion*.

Append hadoop principal to logged queries

Optionally, you can choose to enable appending the Hadoop principal as a comment to the SQL queries that are written to your database logs.

NOTE: This feature only applies if you are connecting to a Hadoop cluster. Otherwise, no value is inserted if this feature is enabled.

Example:

```
execute <unnamed>: SELECT * FROM "public"."artifacts" LIMIT 10 /* <hadoopPrincipal> */
```

In the above, the value of the Hadoop principal is written in the comment.

These types of queries are logged for the following basic activities:

- Data preview: when previewing data from a relational source, a query is executed against the database
- Data import: when selecting a table to import
- Data import using custom SQL:
 - Click Validate button.
 - Custom SQL execution.

This feature enables auditing of Trifacta user activities through your database logs in Hadoop-based environments.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following setting and set it to `true`:

```
"feature.addUserIdToSQLQuery.enabled": false,
```

3. Save your changes and restart the platform.

Enable Driver Logging

Optionally, you can enable the inclusion of log entries from the driver underlying a relational connection.

NOTE: This option applies only to relational connections that rely on CData drivers. Some connections may not support this option.

When you create or edit a relational connection, insert the following as part of the Connect String Options:

```
logfile=STDOUT://;verbosity=5;
```

Log entries are included in the `data-service.log` file is included in the standard Support Bundle. For more information, see *Support Bundle Contents*.

Configure Security

For more information, see *Configure Security for Relational Connections*.

Enable SSO Connections

If you have enabled Kerberos on the Hadoop cluster, you can leverage the Kerberos global keytab to enable SSO connections to relational sources. See *Enable SSO for Relational Connections*.

Type Inference

By default, the platform applies type inferencing to all imported datasources. However, for schematized sources, you may wish to disable type inferencing from the platform instead relying on the types provided from the source.

Tip: You can also toggle the use of type inferencing for individual connections or for individual imported datasets.

For more information, see *Configure Type Inference*.

Enable OAuth 2.0 Connectivity

Some supported relational datastores support authentication using OAuth 2.0.

- For each system to which you want to connect, you must create a client app in the target system. For more information, see *Enable OAuth 2.0 Authentication*.
- For a target system for which you have created a client app, you must create at least one client in the Trifacta application. For more information, see *Create OAuth2 Client*.

Relational Access

Contents:

- *Supported Relational Databases*
 - *Ports*
 - *Enable*
 - *Limitations*
 - *Execution at scale*
 - *Password Encryption Key File*
-

The Trifacta® platform can be configured to access data stored in relational database sources over JDBC protocol. When this connection method is used, individual database tables and views can be imported as datasets.

Supported Relational Databases

The Trifacta platform can natively connect to these relational database platforms. Natively supported versions are the following:

- Oracle 12.1.0.2
- SQL Server 12.0.4
- PostgreSQL 9.3.10
- Teradata 14.10+

NOTE: To enable Teradata connections, you must download and install Teradata drivers first. For more information, see *Enable Teradata Access*.

Additional relational connections can be enabled and configured for the platform. For more information, see *Connection Types*.

Ports

For any relational source to which you are connecting, the Trifacta node must be able to access it through the specified host and port value.

Please contact your database administrator for the host and port information.

Enable

This feature is enabled automatically.

To disable:

To prevent users from connecting to relational datasources for importing datasets and writing results, please complete the following configuration changes:

NOTE: Disabling this feature hides existing relational connections.

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.

2. Locate the following setting:

```
Connectivity feature
```

3. Set this value to `Disabled`.

Disable relational publishing

By default, relational connections are read/write, which means that users can create connections that enable writing back to source databases.

- When this feature is enabled, writeback is enabled for all natively supported relational connection types. See *Connection Types*.
- Depending on the connection type, the Trifacta platform writes its data to different field types in the target database. For more information, see *Type Conversions*.
- Some limitations apply to relational writeback. See Limitations below.

As needed, you can disable this feature.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `false`:

```
"webapp.connectivity.relationalWriteback.enabled": true,
```

3. Save changes and restart the platform.

Publishing through relational connections is disabled.

Limitations

NOTE: Unless otherwise noted, authentication to a relational connection requires basic authentication (username/password) credentials.

- You cannot swap relational sources if they are from databases provided by different vendors. See *Flow View Page*.
- There are some differences in behavior between reading tables and views. See *Using Databases*.

Limitations on relational publishing:

When the relational publishing feature is enabled, it is automatically enabled for all platform-native connection types. You cannot disable relational publishing for Oracle, SQL Server, PostgreSQL, or Teradata connection types. Before you enable, please verify that all user accounts accessing databases of these types have appropriate permissions.

NOTE: Writing back to the database utilizes the same user credentials and therefore permissions as reading from it. Please verify that the users who are creating read/write relational connections have appropriate access.

- You cannot ad-hoc publish to a relational target. Relational publishing is only supported through the Run Job page.
- You write to multiple relational outputs from the same job only if they are from the same vendor.
 - For example, if you have two SQL Server connections A and B, you can write one set of results to A and another set of results to B for the same job.
 - If A and B are from different database vendors, you cannot write to them from the same job.

Execution at scale

Jobs for large-scale relational sources can be executed on the Spark running environment. After the data source has been imported and wrangled, no additional configuration is required to execute at scale.

NOTE: End-to-end performance is likely to be impacted by:

- streaming data volumes over 1 TB from the source,
- streaming from multiple concurrent sources,
- overall network bandwidth.

When the job is completed, any temporary files are automatically removed from HDFS.

For more information, see *Run Job Page*.

Password Encryption Key File

Relational database passwords are encrypted using key files:

- **Passwords in transit:** The platform uses a proprietary encryption key that is invoked each time a relational password is shared among platform services.
- **Passwords at rest:** For creating connections to your relational sources, you must create and reference your own encryption key file. This encryption key is accessing your relational connections from the web application. For more information, see *Create Encryption Key File*.

Enable Custom SQL Query

Contents:

- *Limitations*
 - *Enable*
 - *Enable multi-statement*
 - *Configure query timeout*
 - *Use Custom SQL Queries*
-

To improve performance of your Hive or relational connections, custom SQL queries can be enabled to push the initial filtration of table rows and columns back the database, which is more efficient at performing this task. Instead of loading the entire table into the Trifacta® application and then performing the filtration through the Transformer page, you can insert basic SQL commands as part of your relational queries to collect only the rows and columns of interest from the source.

When enabled, custom SQL query is available for all relational sources.

Limitations

See *Create Dataset with SQL*.

Enable

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following setting:

```
Enable custom SQL Query
```

Setting	Description
enabled	Set to true to enable the SQL pushdown feature. By default, this feature is enabled.

Enable multi-statement

Optionally, you can enable the use of multiple statements in your SQL queries for imported datasets.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following setting:

```
"webapp.connectivity.customSQLQuery.enableMultiStatement": false,
```

Setting	Description
enableMultiStatement	<p>When set to <code>true</code>, you can insert multi-line statements in your SQL pushdown queries. The default is <code>false</code>.</p> <div> <p>NOTE: Use of multi-line SQL has limitations. See <i>Create Dataset with SQL</i>.</p> </div>

3. Save your changes and restart the platform.

Configure query timeout

As needed, you can configure the maximum permitted load time before timeout from the application. See *Configure Application Limits*.

Use Custom SQL Queries

When custom SQL query is enabled, you can enter customized SQL statements in the imported dataset page as part of the import process. See *Import Data Page*.

For examples, see *Create Dataset with SQL*.

After a dataset has been imported using custom SQL, you can edit the SQL as needed. See *Dataset Details Page*.

Configure JDBC Ingestion

Contents:

- *Recommended Table Size*
- *Performance*
- *Enable*
- *Configure*
 - *Configure Ingestion*
 - *Logging*
- *Monitoring Progress*
- *Logging*

This section describes some of the configuration options for the JDBC (relational) ingestion, which support faster execution of JDBC-based jobs.

Data ingestion works by streaming a JDBC source into a temporary storage space in the base storage layer to stage the data for job execution. The job can then be run on Photon or Spark. When the job is complete, the temporary data is removed from base storage or retained in the cache (if it is enabled).

- Data ingestion happens for Spark and Trifacta Photon jobs.
- Data ingestion applies only to JDBC sources that are not native to the running environment. For example, JDBC ingestion is not supported for Hive.
- Schema information is retained from the schematized source and is applied during publication of the generated results.
- Supported for HDFS and other large-scale backend datastores.

Data caching refers to the process of ingesting and storing data sources on the Trifacta node for a period of time for faster access if they are needed for additional platform operations.

Tip: Data ingestion and data caching can work together. For more information on data caching, see *Configure Data Source Caching*.

Job Type	JDBC Ingestion Enabled only	JDBC Ingestion and Caching Enabled
transformation job	Data is retrieved from the source and stored in a temporary backend location for use in sampling.	Data is retrieved from the source for the job and refreshes the cache where applicable.
sampling job	See previous.	<div>Cache is first checked for valid data objects. Outdated objects are retrieved from the data source.</div> <div>Retrieved data refreshes the cache.</div> <div>NOTE: Caching applies only to full scan sampling jobs. Quick scan sampling is performed in the Trifacta Photon running environment.</div> <div>As needed you can force an override of the cache when executing the sample. Data is collected from the source. See <i>Samples Panel</i>.</div>

Recommended Table Size

Although there is no absolute limit, you should avoid executing jobs on tables over several 100 GBs. Larger data sources can significantly impact end-to-end performance.

NOTE: This recommendation applies to all JDBC-based jobs.

Performance

Rule of thumb:

- For a single job with 16 ingest jobs occurring in parallel, maximum expected transfer rate is 1 GB/minute.

Scalability:

- 1 ingest job per source, meaning a dataset with 3 sources = 3 ingest jobs.
- Rule of thumb for max concurrent jobs for a similar edge node:

```
max concurrent sources = max cores - cores used for services
```

- Above is valid until the network becomes a bottleneck. Internally, the above maxed out at about 15 concurrent sources.
- Default concurrent jobs = 16, pool size of 10, 2 minute timeout on pool. This is to prevent overloading of your database.
- Adding more concurrent jobs once network has bottleneck will start slow down all the transfer jobs simultaneously.
- If processing is fully saturated (# of workers is maxed):
 - max transfer can drop to 1/3 GB/minute.
 - Ingest waits for two minutes to acquire a connection. If after two minutes a connection cannot be acquired, the job fails.
- When job is queued for processing:
 - Job is silently queued and appears to be in progress.
 - Service waits until other jobs complete.
 - Currently, there is no timeout for queueing based on the maximum number of concurrent ingest jobs.

Enable

To enable JDBC ingestion and performance caching, the first two of the following parameters must be enabled.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter Name	Description
<code>webapp.connectivity.ingest.enabled</code>	Enables JDBC ingestion. Default is <code>true</code> .
<code>feature.jdbcIngestionCaching.enabled</code>	Enables caching of ingested JDBC data. <div>NOTE: <code>webapp.connectivity.ingest.enabled</code> must be set to <code>true</code> to enable JDBC caching.</div>

	When disabled, no caching of JDBC data sources is performed. For more information on caching, see <i>Configure Data Source Caching</i> .
<code>feature.enableLongLoading</code>	When enabled, you can monitor the ingestion of long-loading JDBC datasets through the Import Data page. Default is <code>true</code> . Tip: After a long-loading dataset has been ingested, importing the data and loading it in the Transformer page should perform faster.
<code>feature.enableParquetLongLoading</code>	When enabled, you can monitor the ingestion of long-loading Parquet datasets. Default is <code>false</code> .
<code>longloading.addToFlow</code>	When long-loading is enabled, set this value to <code>true</code> to enable monitoring of the ingest process when large relational sources are added to a flow. Default is <code>true</code> . See <i>Flow View Page</i> .
<code>longloading.addToLibrary</code>	When long-loading is enabled, this feature enables monitoring of the ingest process when large relational sources are added to the library. Default is <code>true</code> . See <i>Library Page</i> .

Configure

In the following sections, you can review the available configuration parameters for JDBC ingest.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Configure Ingestion

Parameter Name	Description
<code>batchserver.workers.ingest.max</code>	Maximum number of ingester threads that can run on the Trifacta platform at the same time.
<code>batchserver.workers.ingest.bufferSizeBytes</code>	Memory buffer size while copying to backend storage. A larger size for the buffer yields fewer network calls, which in rare cases may speed up ingest.
<code>batch-job-runner.cleanup.enabled</code>	Clean up after job, which deletes the ingested data from backend storage. Default is <code>true</code> . NOTE: If JDBC ingestion is disabled, relational source data is not removed from platform backend storage. This feature can be disabled for debugging and should be re-enabled afterward. NOTE: This setting rarely applies if JDBC ingest caching has been enabled.

Logging

Parameter Name	Description
<code>data-service.systemProperties.logging.level</code>	When the logging level is set to <code>debug</code> , log messages on JDBC caching are recorded in the data service log. NOTE: Use this setting for debug purposes only, as the log files can grow quite large. Lower the setting after the issue has been debugged. See Logging below.

Monitoring Progress

You can use the following methods to track progress of ingestion jobs.

- **Through application:** In the Jobs page, you can track progress of all jobs, including ingestion. Where there are errors, you can download logs for further review.
 - See *Jobs Page*.
 - See Logging below.
- **Through APIs:**
 - You can track status of `jobType=ingest` jobs through the API endpoints.
 - From the above endpoint, get the ingest jobld to track progress.
 - See <https://api.trifacta.com/ee/es.t/index.html#operation/getJobGroup>

Logging

During and after an ingest job, you can download the job logs through the Jobs page. Logs include:

- All details including errors
- Progress on ingest transfer
- Record ingestion

See *Jobs Page*.

Configure Data Source Caching

Contents:

- *Limitations*
 - *Enable*
 - *Configure*
 - *Configure Storage*
 - *Global or user cache*
 - *Cache sizing*
 - *Logging*
-

This section describes some of the configuration options for the data source caching feature. When data is read from the source, the Trifacta® platform can populate a global or user-specific cache with ingested objects. These objects can be sourced from:

- JDBC tables, which are ingested as part of running jobs
- Excel data, which must be converted to CSV format and ingested
- PDF table data, which must be converted to CSV format and ingested

After initial ingest, cached objects can be referenced later for faster performance on tasks such as sampling and job execution.

Limitations

- JDBC ingest caching is not supported for Hive.

Enable

To enable JDBC ingestion and performance caching, the following parameter must be enabled.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter Name	Description
<code>feature.jdbcIngestionCaching.enabled</code>	<p>Enables caching of ingested JDBC data.</p> <div>NOTE: <code>webapp.connectivity.ingest.enabled</code> must be set to <code>true</code> to enable JDBC caching.</div> <p>When disabled, no caching of JDBC data sources is performed.</p>

Configure

In the following sections, you can review the available configuration parameters for performance caching.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Configure Storage

When files are ingested, they are stored in one of the following locations:

- **If caching is enabled:**
 - **If the global datasource cache is enabled:** files are stored in a user-specific sub-folder of the path indicated by the following parameter: `hdfs.pathsConfig.globalDataSourceCache`
 - **If the global cache is disabled:** files are stored in a sub-folder of the output area for each user, named: `/.datasourceCache`.
- **If caching is disabled:** files are stored in a sub-folder within the jobs area for the job group. Ingested files are stored in as `.trifacta` files.

NOTE: Whenever a job is run, its source files must be re-ingested. If two or more datasets in the same job run share the same source, only one copy of the source is ingested.

Additional information is provided below.

Global or user cache

Parameter	Description
<code>datasourceCaching.useGlobalDataSourceCache</code>	<p>When set to <code>true</code>, the platform uses the global data source cache location for storing cached ingest data.</p> <p>NOTE: When global caching is enabled, data is still stored individual locations per user. Through the application, users cannot access the cached objects stored for other users.</p> <p>When set to <code>false</code>, the platform uses the output directory for each user for storing cached ingest data. Within the output directory, cached data is stored in the <code>/.datasourceCache</code> directory.</p> <p>NOTE: You should verify that there is sufficient storage in each user's output directory to store the maximum cache size as well as any projected uploaded datasets.</p>
<code>hdfs.pathsConfig.globalDataSourceCache</code>	<p>Specifies the path of the global datasource cache, if it is enabled. Specify the path from the root folder of the backend datastore.</p> <p>Tip: This setting applies to HDFS or other backend datastores.</p>

Cache sizing

Parameter	Description
<code>datasourceCaching.refreshThreshold</code>	<p>The number of hours that an object can be cached. If the object has not been refreshed in that period of time, the next request for the datasource collects fresh data from the source.</p> <p>By default, this value is set to 168 (one week).</p>
<code>datasourceCaching.maxSize</code>	<p>Maximum size in bytes of the datasource cache. This value applies to individual user caches when either global or user-specific caching is enabled.</p>

Logging

Parameter Name	Description
<code>data-service.systemProperties.logging.level</code>	<p>When the logging level is set to <code>debug</code>, log messages on JDBC caching are recorded in the data service log.</p> <p>NOTE: Use this setting for debug purposes only, as the log files can grow quite large. Lower the setting after the issue has been debugged.</p>

See Logging below.

When the logging level is set to `debug` for the data service and caching is enabled, cache messages are logged. These messages include:

- Cache hits and misses
- Cache key generation

Configure Security for Relational Connections

Contents:

- *User Security*
 - *Connection Security Levels*
 - *Credential Sharing*
- *Technical Security*
 - *Encryption Key File*
 - *SSL*
 - *Configure long load timeout limits*
 - *Enable SSO authentication*
- *Troubleshooting*
 - *Reading or writing over TLS/SSL fails*

You can apply the following Trifacta® platform features to relational connections to ensure compliance with enterprise practices.

NOTE: These security options apply to external relational connections. For more information configuring security for internal connections to the Trifacta databases, see *Enable SSL for Databases*.

User Security

Connection Security Levels

Connection Security Level	Description
Private	Private connections are created by individuals and are by default accessible only to the individual who created them.
Private and shared	Optionally, they can be shared by individuals with other users. <div>NOTE: If needed, credential sharing can be disabled. See below.</div>
Global	Global connections are either created by administrators or are private connections promoted to global by administrators.

Credential Sharing

By default, users are permitted to share credentials through the application. Credentials can be shared in the following ways:

- A user can create a private connection to a relational database. Through the application, this private connection can be shared with other users, so that they can access the creator's datasets.
- When sharing a flow with another user, the owner of the flow can choose to share the credentials that are necessary to connect to the datasets that are the sources of the flow.

As needed, credential sharing can be disabled.

NOTE: If enterprise policy is to disable the sharing of credentials, collaborators may need to be permitted to store their source data in shared locations.

Tip: Credential sharing can be disabled by individual users when they share a connection. The connection is shared, but the new user must provide new credentials to use the connection.

Steps:

To disable credential sharing at the global level:

1. Login to the application as an administrator.
2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Locate the following parameter. Set this property to `false`:

```
"webapp.enableCredentialSharing": true,
```

4. Save your changes and restart the platform.

Technical Security

The following features enhance the security of individual and global relational connections.

Encryption Key File

Relational database passwords are encrypted using key files:

- **Passwords in transit:** The platform uses a proprietary encryption key that is invoked each time a relational password is shared among platform services.
- **Passwords at rest:** For creating connections to your relational sources, you must create and reference your own encryption key file. This encryption key is accessing your relational connections from the web application.

This encryption key file must be created and installed on the Trifacta node. For more information, see *Create Encryption Key File*.

SSL

You can enable SSL for any connection by adding the following string to the Connect String Opts field:

```
?ssl=true;
```

Tip: Some connection windows have a Use SSL checkbox, which also works.

Configure long load timeout limits

For long loading relational sources, a timeout is applied to limit the permitted load time. As needed, you can modify this limit to account for larger load times.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

1. Locate and edit the following parameter:

```
"webapp.connectivity.longLoadTimeoutMillis": 120000,
```

2. Save your changes and restart the platform.

Property	Description
longLoadTimeoutMillis	Max number of milliseconds to wait for a long-loading data source. The default value is 120000 (2 minutes).

For additional relational configuration settings, see *Configure Data Service*.

Enable SSO authentication

Relational connections can be configured to leverage your enterprise Single Sign-On (SSO) infrastructure for authentication. Additional configuration is required. For more information, see *Enable SSO for Relational Connections*.

Troubleshooting

Reading or writing over TLS/SSL fails

Reading or writing over TLS/SSL may fail with an error message in the data service data service log similar to the following:

```
The server selected protocol version TLS11 is not accepted by client preferences [TLS12, SSL20Hello]
```

In this case:

- External libraries referenced by the data service may use TLS/SSL protocols of their own choosing.
- These libraries are included during initialization of the data service.
- The listed protocol (TLSv1.1) is a version of the TLS protocol that is no longer supported.

Solution:

You can configure the platform to override the default protocols supported by Java 8 and to instead use the set of protocols listed in platform configuration.

1. Administrators can apply this configuration change through the *Admin Settings Page* in the application. If the application is not available, the settings are available in `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. When set to `true`, the following parameter instructs the data service to use the protocols listed in Admin Settings page instead. Set this parameter to `true`:

```
"data-service.httpsProtocols.reset": false,
```

Setting	Description
false	(default) Supported HTTPS protocols are defined by Java 8.
true	Supported HTTPS protocols are defined by the Trifacta platform.

3. Locate the following parameter:

```
"data-service.httpsProtocols.defaultProtocols": "SSLv3,TLSv1,TLSv1.1,TLSv1.2"
```

Tip: You can enter any TLS/SSL protocol supported by Java 8 in the above. Other protocols are likely to cause read/write failures.

4. In this case, you can add the missing protocol to the list, as in the following example:

```
"data-service.httpsProtocols.defaultProtocols": "SSLv3,TLSv1,TLSv1.1,TLSv1.2,TLSv1.1"
```

5. Save your changes and restart the platform.

Enable SSO for Relational Connections

Contents:

- *Limitations*
- *Prerequisites*
- *Configure*
 - *Configure JAAS file and path*
 - *JAAS file*
 - *Specify Kerberos configuration file*
 - *Configure vendor definition file*
- *Example Setup*
- *Use*
 - *Sharing*

This section describes how to enable relational connections to leverage your Hadoop Single Sign-On (SSO) infrastructure. When this feature is enabled and properly configured, users can create relational (JDBC) connections that use SSO that you have already configured.

Connections that were created before this feature is enabled continue to operate as expected without modification.

Limitations

- For this release, this feature applies to SQL Server connections only.
- Cross-realm is not supported. As a result, the SQL Server instance, service principal, and Trifacta® principal must be in the same Kerberos realm.

Prerequisites

- **Kerberos SSO:** You must set up SSO authentication to the Hadoop cluster using Kerberos. This feature uses the global Kerberos keytab. For more information, see *Configure for Kerberos Integration*.

Configure

Configure JAAS file and path

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter	Description
<code>webapp.connectivity.kerberosDelegateConfigPath</code>	<p>Path on the Trifacta node to the location of the JAAS configuration file required by the DataDirect driver.</p> <div>NOTE: The default location is listed below. You may wish to move this file to a location outside of the Trifacta installation to ensure that the file is not overwritten during upgrades.</div> <p>More information on this file is provided below.</p>

JAAS file

For connections that support Kerberos-delegated authentication, the underlying driver supports a JAAS file in which you can provide environment-specific configuration to the driver. As needed, you can modify this file.

Connection Type	Default path to JAAS file
SQL Server	<code>%(topOfTree)s/services/data-service/build/conf/kerberosdelegate.config</code>

Example JAAS file for SQL Server

Below is an example file, where you must apply the Kerberos global keytab and principal values that are to be used to authenticate to use the Kerberos-delegated connections of this type:

where:

- `keytab` = the absolute path on the Trifacta node where the Kerberos global keytab is located.
- `principal` = Set to the service principal name of the user's service account in LDAP.

Specify Kerberos configuration file

On the Trifacta node, locate the following file:

```
<root>/etc/krb5.conf
```

If it doesn't exist, create it with the following content, some of which you must specify:

Setting	Description
<code>default_realm</code>	Set this value to your default Kerberos realm.

forwardable	This value must be set to <code>true</code> .
kdc	For each realm that you create, you must create an entry in <code>[realms]</code> . For the <code>kdc</code> entry, apply the KDC domain that the JDBC connection should use.
my_domain	For each domain to which the Kerberos delegation applies, you must create an entry in <code>[domain_realm]</code> . Entries should look like the following: <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> example.com = EXAMPLE.COM </div>

Modify the location of the Kerberos configuration file

If you need to move the location of the file from the default one, please complete the following:

Steps:

1. If you haven't already done so, copy the file from its current location to its preferred location.
2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Specify the path to the new location in the following parameter:

```
"webapp.connectivity.krb5Path": "/etc/krb5.conf";
```

4. Save your changes.

Configure vendor definition file

For each vendor that supports SSO connections you must modify a setting in a configuration file on the Trifacta node. This change can only be applied for vendors that support Kerberized SSO connections.

Steps:

1. On the Trifacta node, navigate to the following directory:

```
/opt/trifacta/services/data-service/build/conf/vendor
```

2. In the `vendor` directory, each JDBC vendor has a sub-directory. Open the vendor directory.
3. Edit `connection-metadata.json`.
4. Locate the `credentialType` property. Set the value to `kerberosDelegate`.
5. Save your changes and restart the platform.
6. When you create your connection, select `kerberosDelegate` from the Credential Type drop-down.

Example Setup

The following example uses the default Kerberos realm to set an SSO connection to a SQL Server instance. This example is intended to demonstrate one way in which you can set up your SSO connections.

Steps:

1. Create the Trifacta service principal:
 - a. Form: `HTTP/serviceprincipal@REAM`

- b. Enable this flag: `ok_to_auth_as_delegate`
- c. Example:

```
kadmin -q "addprinc -randkey +ok_to_auth_as_delegate HTTP/serviceprincipal"  
kadmin -q "addprinc -randkey +ok_to_auth_as_delegate HTTP/serviceprincipal@REALM"
```

- d. For more information on delegation flags, see https://web.mit.edu/kerberos/krb5-1.12/doc/admin/admin_commands/kadmin_local.html
2. Generate a keytab for the Trifacta service principal.
 3. Register the Trifacta service principal for Microsoft Sql Server instance:
 - a. Enable this flag: `ok_as_delegated`
 - b. Example:

```
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>:<port>"  
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>:<port>@REALM"  
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>"  
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>@REALM"
```

- c. For more information on setting this flag, see <https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/register-a-service-principal-name-for-kerberos-connections?view=sql-server-2017>
4. Create a linked SQL Server account:
 - a. Account must have the same name as the end-user principal.
 - b. Account needs connect permissions at least.

NOTE: If you are using LDAP/AD SSO, you can register all of the above SPNs using AD mechanisms. You do not have to use the delegation flags. Delegation can be managed through the UI for the service account.

Use

When you create a new connection of a supported type, you can select the Kerberos Delegate credentials type. When selected, no username or credentials are applied as part of the connection object. Instead, authentication is determined via Kerberos authentication with the cluster.

- *Microsoft SQL Server Connections*

Sharing

When sharing SSO connections, the credentials for the connection cannot be shared for security reasons. The Kerberos principal for the user with whom the connection is shared is applied. That user must have the appropriate permissions to access any required data through the connection. See *Overview of Sharing*.

Enable OAuth 2.0 Authentication

Contents:

- *Enable*
 - *Enable OAuth 2.0 client creation*
 - *Configure host URL*
 - *Enable Secure Token Service*
 - *Install Secure Token Database*
 - *Create OAuth 2.0 App*
 - *Create OAuth 2.0 Client*
 - *Authenticate OAuth 2.0 Connections*
-

Workspace administrators can enable the use of OAuth 2.0 authentication for creating connections to third-party datastores that support OAuth 2.0 or greater authentication.

OAuth 2.0 is an industry-standard protocol for authorization between systems. In Trifacta®, it is implemented as a security protocol for access to data sources and publishing destinations. Trifacta administrators can enable users of the product to connect to specified third-party systems through an **OAuth 2.0 client app** that you create in the system, using an **OAuth 2.0 client** reference that is created in the Trifacta application.

When enabled and configured, the Trifacta application uses the OAuth 2.0 client to create a **secure token**, which is used to authenticate to the third-party system. Internally, the Trifacta platform leverages the **secure token service** to manage the creation and use of these secure tokens. For OAuth 2.0, this service uses a backing database for storing tokens. **Requirements:**

- **OAuth 2.0 client app:** In the target system, you must create an object called a **client app**, which provides an authentication interface into the system for external connections.
 - You must create one client app for each external system to which you are enabling connectivity.
- **OAuth 2.0 client:** In the Trifacta application, you must create at least one configuration object for each client app that you have created.
- Enable the creation of OAuth 2.0 clients in the Trifacta application.
- Enable the secure token service, which is used to manage the secure tokens of the Trifacta application.
- Install and configure the database used by the secure token service. Installation should happen automatically as part of the normal install or upgrade process.

Details on these requirements are listed below.

Enable

Enable OAuth 2.0 client creation

The ability to create OAuth 2.0 clients in the Trifacta application must be enabled. Please verify the following configuration.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Please locate the following setting and set it to `true`:

```
"feature.adminConsole.oauth2ClientsManagement.enabled": true,
```

3. Save your changes.

Configure host URL

When you create an OAuth 2.0 connection, the connection object must pass to the client on the target platform the URL of the Trifacta application, so that the client can re-direct queries back to the application after authentication is complete.

Please verify that the following parameter is set to the public value of the host and port number of the Trifacta application. It should be in the following form:

```
<http/https>://<host>:<port>
```

where:

- `<http/https>` = protocol to use to connect
- `<host>` = host name for external users to access the application
- `<port>` = port number for external users to access the application. Typically, this value is 3005.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Please verify the following setting is set to the correct value for your environment:

```
"webapp.hostUrl": "https://www.trifacta.example.com:3005",
```

3. Save your changes.

Enable Secure Token Service

OAuth 2.0 requires the use of the secure token service for managing the authentication tokens. For more information, see *Configure Secure Token Service*.

Install Secure Token Database

The secure token service database is installed as part of normal database install or upgrade operations. For more information, see *Install Databases*.

Create OAuth 2.0 App

For each target system, you must create an OAuth 2.0 app in the system, which provides an external interface for Trifacta.

NOTE: The requirements for creating an OAuth 2.0 app depend on the system. Some example setups are available below. For more information, please see the documentation provided with your target system.

Create OAuth 2.0 Client

Through the Trifacta application, you must create an OAuth 2.0 client that connects to the OAuth 2.0 app that you have created.

- In the Admin console, select **OAuth 2.0 Clients**. For more information, see *OAuth 2.0 Clients Page*.

- For more information on creating a client, see *Create OAuth2 Client*.

Authenticate OAuth 2.0 Connections

When you create a connection that uses OAuth 2.0, the specified connection must be authorized to be given access to the datastore. In the Create Connection window, click **Authenticate**.

NOTE: If you modify a connection or the tokens generated under the previous authorization have expired, you must re-authenticate the connection. Edit the connection and click **Re-authenticate**.

Create OAuth2 Client

Contents:

- *Prerequisites*
 - *Enable*
 - *Create OAuth 2.0 App*
 - *Configure*
-

Through the Trifacta® application, workspace administrators can configure OAuth 2.0 clients to enable connectivity to third-party datastores that support OAuth 2.0 or greater authentication. In the OAuth 2.0 Clients page, click **Register OAuth 2.0 Client**.

Prerequisites

Enable

OAuth 2.0 authentication must be enabled in the Trifacta platform. For more information, see *Enable OAuth 2.0 Authentication*.

Create OAuth 2.0 App

Before you create an OAuth Client in Trifacta, you must create a corresponding Client App in the system with which you are integrating.

Configure

Specify the following properties for your OAuth 2.0 client.

Property	Description
Type	Select the type of client from the drop-down list.
Name	Display name of your OAuth 2.0 client.
Client ID	The client identifier for the OAuth 2.0 app that you created.
Client Secret	The client secret for the OAuth 2.0 app that you created.
Authorization URL	The URL that is used for authorizing to the client app.
Token URL	The token URL for the client app that you created.
Scopes	Scopes are space-delimited strings that can be used to pass parameters to the client app that you created. <div>NOTE: The specific scopes that you can pass depends on the system with which you are integrating.</div>
Access Token Expires In	Number of milliseconds that an access token is permitted to be used to connect to the target OAuth 2.0 app. This value must be set to an integer greater than 0. For more information, please see the documentation for your target system.
Refresh Token Expires In	Number of milliseconds of inactivity that are permitted before an access token is expired. <div>Tip: To create non-expiring tokens, set this value to 0.</div>

For more information, please see the documentation for your target system.

OAuth 2.0 for Google Analytics

Contents:

- *Prerequisites*
 - *Create OAuth 2.0 Client App for Google Analytics*
 - *Enable external user in project*
 - *Create OAuth 2.0 credentials*
 - *Enable API access*
 - *Create OAuth 2.0 Client for Google Analytics*
 - *Create Google Analytics Connection*
-

This section describes the steps to configure the Trifacta® application to integrate with Google Analytics using OAuth 2.0 to authenticate.

Prerequisites

- OAuth 2.0 authentication must be enabled in the Trifacta platform.
- An OAuth 2.0 client is required for Trifacta Self-Managed Enterprise Edition only.
- For more information, see *Enable OAuth 2.0 Authentication*.

Create OAuth 2.0 Client App for Google Analytics

Enable external user in project

You must enable external access to the project containing your Google Analytics data.

NOTE: This step configures access through the consent screen for your project. If you have previously completed this step for the project, you can skip this section.

Steps:

1. Navigate to the Google Console for your project: <https://console.cloud.google.com/>.
2. From the left menu, select **APIs & Services > OAuth consent screen**.
3. For User Type, select **External**.
4. Click **Create**.
5. You can provide a logo and name for this client. For example:

Tip: You can use your own logo and product name if preferred.

- a. Right-click the logo in the Trifacta application and download it to your desktop. Right-click the image and select **Save As....** Upload it to the consent screen.
 - b. The name of the product can be: Trifacta Self-Managed Enterprise Edition.
6. Do not add Scopes or Test Users.
 7. Save your changes.

Create OAuth 2.0 credentials

You must create a set of credentials to use when accessing your Google project.

Steps:

1. From the APIs & Services menu, select **Credentials**.
2. At the top of the screen, click **+CREATE CREDENTIALS**.
3. Select **OAuth client Id**.
4. For Application type, select **Web application**.
5. Fill the values for the following settings:

Setting	Value
Name	Provide a descriptive name. Example: Google_Analytics
Authorized JavaScript origins	Do not add a value for this setting.
Authorized Redirect URIs	Set the value to the following: <div>https://<login_url>:<port_number>/oauth2/callback</div>

6. Click **Create**.
7. Retain the values for ClientId and Client Secret. These values must be applied in the Trifacta application.

Enable API access

You must enable access to Google Analytics APIs through your project. You can enable one or more of the APIs listed below.

- Google Analytics API: <https://console.cloud.google.com/apis/library/analytics.googleapis.com>
- Google Analytics Reporting API: <https://console.cloud.google.com/apis/library/analyticsreporting.googleapis.com>

Steps:

1. Navigate to listed URL.
2. Click **Enable**.

Create OAuth 2.0 Client for Google Analytics

After the Google Analytics app is created, you must create an OAuth 2.0 client in the Trifacta application, which is used to integrate with the OAuth 2.0 connected app that you created above.

NOTE: You must create one OAuth 2.0 client in the Trifacta application for each Google Analytics connected app that you wish to use.

Steps:

1. Login to the Trifacta application as a workspace administrator.
2. In the lefthand menu, select **User menu > Admin console > OAuth2.0 Clients**.
3. In the OAuth2.0 Clients page, click **Register OAuth2.0 Client**.
4. Specify the new client. Apply the following values:

Setting	Description
Type	Set to google_analytics.
Name	Display name for the OAuth 2.0 client in the Trifacta application.
Client ID	Set this value to the Client Id value that you retained from your Google Analytics app.

Client Secret	Set this value to the Client Secret value that you retained from your Google Analytics app.
Authorization URL	Set this value to the following: <code>https://accounts.google.com/o/oauth2/v2/auth</code>
Token URL	Set this value to the following: <code>https://oauth2.googleapis.com/token</code>
Scopes	Please insert the following value: <code>https://www.googleapis.com/auth/drive.readonly</code>
Access Token Expires In	Set this value (in milliseconds) to 3600000 (1 hour).
Refresh Token Expires In	Set the value to 0 (does not expire).

5. To save your OAuth 2.0 client, click **Save**.

For more information, see *Create OAuth2 Client*.

Create Google Analytics Connection

After you have created the two OAuth 2.0 client references, you can create a connection to your Google Analytics data.

NOTE: You must create a separate connection for each OAuth 2.0 client that is available in the Trifacta application.

For more information, see *Google Analytics Connections*.

OAuth 2.0 for Google Sheets

Contents:

- *Prerequisites*
 - *Create OAuth 2.0 Client App for Google Sheets*
 - *Enable external user in project*
 - *Create OAuth 2.0 credentials*
 - *Enable API access*
 - *Create OAuth 2.0 Client for Google Sheets*
 - *Create Google Sheets Connection*
-

This section describes the steps to configure the Trifacta® application to integrate with Google Sheets using OAuth 2.0 to authenticate.

Prerequisites

- OAuth 2.0 authentication must be enabled in the Trifacta platform.
- An OAuth 2.0 client is required for Trifacta Self-Managed Enterprise Edition only.
- For more information, see *Enable OAuth 2.0 Authentication*.

Create OAuth 2.0 Client App for Google Sheets

Enable external user in project

You must enable external access to the project containing your Google Sheets data.

NOTE: This step configures access through the consent screen for your project. If you have already done this step for the project, you can skip this section.

Steps:

1. Navigate to the Google Console for your project: <https://console.cloud.google.com/>.
2. From the left menu, select **APIs & Services > OAuth consent screen**.
3. For User Type, select **External**.
4. Click **Create**.
5. You can provide a logo and name for this client. For example:

Tip: You can use your own logo and product name if preferred.

- a. Right-click the logo in the Trifacta application and download it to your desktop. Right-click the image and select **Save As....** Upload it to the consent screen.
 - b. The name of the product can be: Trifacta Self-Managed Enterprise Edition.
6. Do not add Scopes or Test Users.
 7. Save your changes.

Create OAuth 2.0 credentials

You must create a set of credentials to use when accessing your Google project.

Steps:

1. From the APIs & Services menu, select **Credentials**.
2. At the top of the screen, click **+CREATE CREDENTIALS**.
3. Select **OAuth client Id**.
4. For Application type, select **Web application**.
5. Fill the values for the following settings:

Setting	Value
Name	Provide a descriptive name. Example: Google_Analytics
Authorized JavaScript origins	Do not add a value for this setting.
Authorized Redirect URIs	Set the value to the following: <div> <a href="https://<login_url>:<port_number>/oauth2/callback">https://<login_url>:<port_number>/oauth2/callback </div>

6. Click **Create**.
7. Retain the values for ClientId and Client Secret. These values must be applied in the Trifacta application.

Enable API access

You must enable API access to your project.

Steps:

1. To enable the Google Sheets API, navigate to the following URL:
<https://console.cloud.google.com/apis/library/sheets.googleapis.com>
2. Click **Enable**.
3. To use Google Sheets, you must also enable the Google Drive API. Navigate to the following URL:
<https://console.cloud.google.com/apis/library/drive.googleapis.com>
4. Click **Enable**.

Create OAuth 2.0 Client for Google Sheets

After the Google Sheets app is created, you must create an OAuth 2.0 client in the Trifacta application, which is used to integrate with the OAuth 2.0 connected app that you created above.

NOTE: You must create one OAuth 2.0 client in the Trifacta application for each Google Sheets connected app that you wish to use.

Steps:

1. Login to the Trifacta application as a workspace administrator.
2. In the lefthand menu, select **User menu > Admin console > OAuth2.0 Clients**.
3. In the OAuth2.0 Clients page, click **Register OAuth2.0 Client**.
4. Specify the new client. Apply the following values:

Setting	Description
Type	Set to <code>google_sheets</code> .
Name	Display name for the OAuth 2.0 client in the Trifacta application.
Client ID	Set this value to the Client Id value that you retained from your Google Sheets app.
Client Secret	Set this value to the Client Secret value that you retained from your Google Sheets app.

Authorization URL	Set this value to the following: <div>https://accounts.google.com/o/oauth2/v2/auth</div>
Token URL	Set this value to the following: <div>https://oauth2.googleapis.com/token</div>
Scopes	Please insert the following value: <div>https://www.googleapis.com/auth/drive.readonly</div>
Access Token Expires In	Set this value (in milliseconds) to 3600000 (1 hour).
Refresh Token Expires In	Set the value to 0 (does not expire).

5. To save your OAuth 2.0 client, click **Save**.

For more information, see *Create OAuth2 Client*.

Create Google Sheets Connection

After you have created the two OAuth 2.0 client references, you can create a connection to your Google Sheets data.

NOTE: You must create a separate connection for each OAuth 2.0 client that is available in the Trifacta application.

For more information, see *Google Sheets Connections*.

OAuth 2.0 for Salesforce

Contents:

- *Prerequisites*
 - *Create OAuth 2.0 Client App in Salesforce*
 - *Scopes for Salesforce*
 - *Create OAuth 2.0 Client for Salesforce*
 - *Create Salesforce Connection*
-

This section describes the steps to configure the Trifacta® application to integrate with your Salesforce deployment using OAuth 2.0 to authenticate.

Prerequisites

OAuth 2.0 authentication must be enabled in the Trifacta platform. For more information, see *Enable OAuth 2.0 Authentication*.

Create OAuth 2.0 Client App in Salesforce

In Salesforce, you must create the connected app through which the Trifacta application uses OAuth 2.0 to access and connect to your Salesforce data.

Steps:

1. **Login:** Log in to the Salesforce account in which you want the OAuth 2.0 app to be created.
2. In the top bar, click **Setup**.
3. In the left nav bar, search for: `apps`. Then, navigate to **Create > Apps**.
4. **Create connected app:** In the Connected Apps section, click **New**.
 - a. To create a connected app, please complete the listed fields with the appropriate information. Some specifics:

Field	Description
Connected App Name	Display name of the app. Suggested: Trifacta application
API Name	Please add the value for Connected App Name here.
Contact Email	Add a valid contact email address.
Logo image URL	(optional) Upload an app logo as needed.
Enable OAuth Settings	Select this option.
Callback URL	<div>Please provide a URL in the following format:<div>https://<platform_login_url>/oauth2/callback</div> where: <platform_login_url> = the URL that is accessed to log in to Trifacta platform. This value may or may not include a port number.</div>
Selected OAuth Scopes	Please select the following scopes:1. <code>api2.refresh_token</code>

Require secret for web server flow	Select this option.
------------------------------------	---------------------

- b. At the bottom of the screen, click **Save** to save the connected app.
5. **Configure policies:** In the left nav bar, select **Manage > Connected apps**.
 - a. Then, click the **Edit Policies** button.
 - b. In the Edit Policies screen, click the **Manage** button.
 - c. Under Session Policies, select the Timeout Value. Set this value to 24 hours.
 - d. Click **Save** to save your connected app.
6. **Retain values:** Your Salesforce connected app configuration is complete. Please acquire the following information from the app listing in Salesforce. These parameter values are needed for creating the OAuth 2.0 client in the Trifacta application:

Parameter	Description
Consumer Key	This value is used as the Client Id in Trifacta application. Select Click to reveal to display.
Consumer Secret	This value is used as the Client Secret in Trifacta application. Select Click to reveal to display.
Selected OAuth Scopes	Acquire this values. Unless otherwise specified, these values should include:1. api2. refresh_token
Access token expires in	Navigate to Manage > Edit Policies . Typically, this value in milliseconds is set to 1 hour (3600000 milliseconds). For more information, see https://help.salesforce.com/articleView?id=connected_app_manage_session_policies.htm&type=5 .

7. Save any changes to the connected app.

Scopes for Salesforce

The following scopes are required in the connected app for the Trifacta application to access Salesforce:

Scope	Description
api	(required) Provides REST API access to Salesforce.
refresh_token	(required) This token allows the OAuth 2.0 client to refresh the connection with Salesforce without user interaction.

Create OAuth 2.0 Client for Salesforce

After the Salesforce connected app is created, you must create an OAuth 2.0 client in the Trifacta application, which is used to integrate with the OAuth 2.0 connected app that you created above.

NOTE: You must create one OAuth 2.0 client in the Trifacta application for each Salesforce connected app that you wish to use.

Steps:

1. Login to the Trifacta application as a workspace administrator.
2. In the lefthand menu, select **User menu > Admin console > OAuth2.0 Clients**.
3. In the OAuth2.0 Clients page, click **Register OAuth2.0 Client**.
4. Specify the new client. Apply the following values:

Setting	Description
Type	Set to <code>salesforce</code> .
Name	Display name for the OAuth 2.0 client in the Trifacta application.
Client ID	Set this value to the Consumer Key value in your Salesforce connected app.

Client Secret	Set this value to the Consumer Secret value in your Salesforce connected app.
Authorization URL	Set this value to the following: <code>https://login.salesforce.com/services/oauth2/authorize</code>
Token URL	Set this value to the following: <code>https://login.salesforce.com/services/oauth2/token</code>
Scopes	Insert the scopes you specified as a space-separated list.
Access Token Expires In	Set this value to the corresponding value in your Salesforce connected app. See above.
Refresh Token Expires In	Set this value to the number of milliseconds after which the refresh token expires. Set the value to 0 (does not expire).

5. To save your OAuth 2.0 client, click **Save**.

For more information, see *Create OAuth2 Client*.

Create Salesforce Connection

After you have created the two OAuth 2.0 client references, you can create a connection to your Salesforce data.

NOTE: You must create a separate connection for each OAuth 2.0 client that is available in the Trifacta application.

For more information, see *Salesforce Connections*.

OAuth 2.0 for Snowflake

Contents:

- *Prerequisites*
- *Create OAuth 2.0 Client App in Snowflake*
- *Create OAuth 2.0 Client for Snowflake*
 - *Scopes for Snowflake*
- *Create Snowflake Connection*
- *Troubleshooting*
 - *"Invalid consent request" error*

Configure the Trifacta® application to integrate with your Snowflake deployment using OAuth 2.0 to authenticate.

Prerequisites

OAuth 2.0 authentication must be enabled in the Trifacta platform. For more information, see [Enable OAuth 2.0 Authentication](#).

Create OAuth 2.0 Client App in Snowflake

In your Snowflake console, you must create the client app, which includes execution of several SQL statements.

NOTE: You must have the ACCOUNTADMIN role to create the client app.

In Snowflake, this object is called a **security integration**. For more information, see <https://docs.snowflake.com/en/sql-reference/sql/create-security-integration.html>.

Steps:

1. Login to the Snowflake console as an account admin.
2. Click **Worksheets**.
3. For your role, select **ACCOUNTADMIN**.
4. Paste the following command in the worksheet and modify its parameters:

```
CREATE [ OR REPLACE ] SECURITY INTEGRATION [IF NOT EXISTS]
  <NAME>
  TYPE = OAUTH
  OAUTH_CLIENT = CUSTOM
  OAUTH_CLIENT_TYPE = 'CONFIDENTIAL'
  OAUTH_REDIRECT_URI = '<URI>'
  ENABLED = TRUE
  OAUTH_ALLOW_NON_TLS_REDIRECT_URI = FALSE
  [PRE_AUTHORIZED_ROLES_LIST = ( '<role_name_1>' [ , '<role_name_2>' , ... ] ) ]
  [ BLOCKED_ROLES_LIST = ( '<role_name_3>' [ , '<role_name_4>' , ... ] ) ]
  OAUTH_ISSUE_REFRESH_TOKENS = TRUE
  OAUTH_REFRESH_TOKEN_VALIDITY = 7776000 (90 Days)
  [ NETWORK_POLICY = '<network_policy>' ]
  [ COMMENT = '<Description of your Integration>' ]
```

Parameter	Description

<NAME>	Name of the integration. Example: OAuth 2.0 Client
<URI>	Callback URI of the Trifacta platform.
PRE_AUTHORIZED_ROLES_LIST	<p>A comma-separated list of Snowflake roles that do not need user consent when accessing Snowflake. The roles SECURITYADMIN and ACCOUNTADMIN cannot be included in this list.</p> <div> <p>Tip: The roles in this list should match up with the roles that are scoped in the OAuth 2.0 client in the Trifacta application. In the client, you can specify the Snowflake roles that are permitted to use the client for authentication. Roles that are scoped for access that are not in this list must consent to access Snowflake after login. In some use cases, such as API access or scheduled executions, this can be problematic.</p> </div>
BLOCKED_ROLES_LIST	A comma-separated list of Snowflake roles that cannot explicitly consent to use when accessing Snowflake. The roles SECURITYADMIN and ACCOUNTADMIN are included by default in this list. If you need to remove either of those roles, please contact Snowflake Support.
<NETWORK_POLICY>	(Optional) Provide the identifier for any applicable Snowflake network policy.
<COMMENT>	(Optional) Add a comment if needed.

- Run the above command. The security integration is created.
- Paste the following command and run it to acquire the following information: Client ID, Authorization URL, Token URL, and Refresh Token Expires In, where <NAME> is the name you provided above:

```
DESC integration <NAME>
```

Retain the values for the following parameters. You must apply these parameters to the OAuth 2.0 client that you create in the Trifacta application:

Snowflake parameter	Trifacta application Client parameter
OAuth_Client_Id	Client Id
OAuth_Authorization_Endpoint	Authorization URL
OAuth_Token_Endpoint	Token URL
OAuth_Refresh_Token_Validity	Refresh Token Expires In

- Paste the following command and run it to acquire the client secret, where <NAME> is the name you provided above:

```
SELECT SYSTEM$SHOW_OAUTH_CLIENT_SECRETS(' <NAME> ')
```

Retain the values for the following. You must apply these parameters to the OAuth 2.0 client that you create in the Trifacta application:

Snowflake parameter	Trifacta application Client parameter
OAuth_Client_Secret	Client Secret

- Save your changes.

Create OAuth 2.0 Client for Snowflake

After the Snowflake client app is created, you must create an OAuth 2.0 client in the Trifacta application, which is used to integrate with the OAuth 2.0 Client app (security integration) that you created above.

NOTE: You must create one OAuth 2.0 client in the Trifacta application for each Snowflake role that you wish to use. See "Scopes" below for more information.

Steps:

1. Login to the Trifacta application as a workspace administrator.
2. In the lefthand menu, select **User menu > Admin console > OAuth 2.0 Clients**.
3. In the OAuth 2.0 Clients page, click **Register OAuth 2.0.0 Client**.
4. Specify the new client.
 - a. For the Type value, select `snowflake`.
 - b. You must apply the values listed in the previous section to your client object.
 - c. For more information on Scopes, see "Scopes for Snowflake" below.
 - d. Access Token Expires in: 600000

NOTE: The value of 600000 is required for Snowflake.

5. To save your OAuth 2.0 client, click **Save**.

For more information, see *Create OAuth2 Client*.

Scopes for Snowflake

Scopes are space-delimited strings that are passed from the client to the client app as part of the authentication process.

The following scope must be specified as part of your Snowflake client definition:

```
refresh_token session:role:<role_name>
```

Scope	Description
refresh_token	(required) Snowflake session tokens have a short duration. By adding this scope, a refresh token is issued for the session. This token allows the OAuth 2.0 client to refresh the connection with Snowflake without user interaction.
role:<role_name>	<p>(optional) The Snowflake role for which you wish to access its databases, schemas, and tables. If this value is not provided, then the default role is used.</p> <p>NOTE: Only one role can be specified per client. This role must provide access to the databases, schemas, and objects that you wish to make accessible through this client.</p> <p>NOTE: The value for <role_name> is case-sensitive, unless you specified the role in quotes when creating it. For more information, see https://docs.snowflake.com/en/user-guide/oauth-custom.html#scope.</p>

Create Snowflake Connection

After you have created the two OAuth 2.0 client references, you can create a connection to your Snowflake databases.

NOTE: You must create a separate connection for each OAuth 2.0 client that is available in the Trifacta application.

For more information, see *Snowflake Connections*.

Troubleshooting

The following may occur when trying to connect to Snowflake databases using OAuth 2.0.

"Invalid consent request" error

If you receive an invalid consent request error, then the user that is passed for OAuth 2.0 authorization does not have access to the role that is referenced in the corresponding OAuth 2.0 client that you created in the Trifacta application.

You can do one of the following:

- Specify a different user in the connection.
- Create a new OAuth 2.0 client in the Trifacta application which is scoped for a role that the database user has.

NOTE: This new role must also be authorized to use the security integration within Snowflake.

Configure Connectivity for Amazon RDS

This section outlines additional configuration that may be required to create connections between Trifacta® and your relational instances hosted on Amazon RDS.

Configure SSL Connections

You can configure your connection to use SSL for interactions with your Amazon RDS database.

Database server configuration: On the database side, you must disable SSL certificate validation. For more information, see <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html>.

Trifacta application: Since you cannot upload your own SSL certificate to Trifacta, you must perform the following steps to disable SSL validation. These steps must be applied to any database connection in the Trifacta application to an Amazon RDS database instance.

Steps:

1. In the Connections page, select the Amazon RDS database connection that you wish to modify.
2. In the Edit Connection window:
 - a. Select the Enable SSL checkbox.
 - b. For the Connect String Options:
 - i. Find the appropriate connect string to disable client certification validation. See the documentation that was provided with your database distribution.
 - ii. Add that value to any existing value for the Connect String Options.
 - c. Test the connection.
3. Save your changes.

When the above is performed, the Trifacta application is no longer expecting a valid certificate and can use SSL to communicate.

Configure Type Inference

Contents:

- *Configure Type Inference for Schematized Sources*
 - *Enable*
 - *Configure Load Limits for Inference*
 - *Use*
 - *Define for individual connections*
 - *Specify on dataset import*
 - *Configure Type Inference in the Data Grid*
 - *Type Inference on Export*
-

By default, the Trifacta® platform applies its own type inference to datasets when they are imported and again when new steps are applied to the data. This section provides information on how you can configure where type inference is applied in the platform.

Data types are inferred by the Trifacta platform when:

- Imported datasets are originally loaded.
- A new transformation step is added in a recipe.
- Non-inferred types are imported as String type.

Tip: You can use the Change Column Type transformation to override the data type inferred for a column. However, if a new transformation step is added, the column data type is re-inferred, which may override your specific typing. You should consider applying Change Column Type transformations as late as possible in your recipes.

For more information on how the Trifacta platform applies data types to specific sources of data on import, see *Type Conversions*.

Configure Type Inference for Schematized Sources

Optionally, you can choose to disable type inference for schematized sources. A **schematized source** includes column data type information as part of the object definition. The following schematized sources are supported for import into the Trifacta platform:

- All JDBC sources

NOTE: You cannot disable type inference for Oracle sources. This is a known issue.

- Hive
- Redshift

- Avro file format

Enable

Type inference on schematized sources	Setting	Behavior
Enabled	"webapp.connectivity.disableRelationalTypeInference": false,	<p>All imported datasets from schematized sources are automatically inferred by the type system in the Trifacta platform.</p> <p>The inferred data types may be different from those in the source. When the dataset is loaded, data types can be applied to individual columns through the application.</p> <p>Users can apply overrides for:</p> <ul style="list-style-type: none">• Individual connections• Individual datasets at time of import
Disabled	"webapp.connectivity.disableRelationalTypeInference": true,	<p>For schematized data sources, type inference is not automatically inferred by Trifacta platform.</p> <p>Data type information is taken from the source schema and applied where applicable to the dataset. If there is no corresponding data type in the Trifacta platform, the data is imported as String type.</p> <p>Users can apply overrides for:</p> <ul style="list-style-type: none">• Individual connections• Individual datasets at time of import

Please perform the following configuration change to disable type inference of schematized sources at the global level.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Change the following configuration setting to `true`:

```
"webapp.connectivity.disableRelationalTypeInference": false,
```

3. Save your changes.

Configure Load Limits for Inference

When a dataset is imported into the Trifacta application, a volume of data is read from the source, up to the parameterized limits below. These limits define the maximum size of the data read for:

- **Split row inference:** data read for determining where each row ends in the dataset.
- **Type inference:** data read for determining the data types of each column.

Tip: You can raise these limits gradually if you are noticing issues with either data inference or row splits. Raising these values significantly can impact load performance in the Transformer page.

Parameter	Description
-----------	-------------

webapp. loadLimitForSplitInference	Maximum number of bytes to be read from an imported dataset for initial inference for splitting rows. Default value is 20000.
webapp. loadLimitForTypeInference	Maximum number of bytes to be read from an imported dataset for initial inference of column data types. Default value is 524288.

Use

In the application, type inference can be applied to your imported data through the following mechanisms.

Define for individual connections

You can specify individual connections to apply or not apply Trifacta type inference when the connection is created or edited.

NOTE: When Default Column Data Type Inference is disabled for an individual connection, Trifacta type inference can still be applied on import of individual datasets.

For more information, see *Create Connection Window*.

Specify on dataset import

When type inference has been disabled globally for schematized sources, you can choose to enable or disable it for individual source import.

Tip: To compare how data types are imported from the schematized source or when applied by the Trifacta platform, you can import the same schematized source twice. The first instance of the source can be imported with type inference enabled, and the second can be imported with it disabled.

In the Import Data page, click **Edit Settings** on the data source card.

For more information, see *Import Data Page*.

Configure Type Inference in the Data Grid

Type inference is automatically enabled in the data grid. It cannot be disabled.

Tip: You can override the Trifacta data type by applying a Change Column Type transformation.

When a new transformation step is applied, each column is re-inferred for its Trifacta data type.

Type Inference on Export

When you generate results, the current data types in the data grid are applied to the generated results.

If the publishing destination is a schematized environment, the generated results are written to the target environment based on the environment type. These data type mappings cannot be modified.

For more information on output types, see *Type Conversions*.

Troubleshooting Relational Connections

Contents:

- *Problem - Unable to access customer encryption key*
 - *Solution*
 - *Problem - Retrieving sample data for large relational tables is very slow*
 - *Solution*
-

Problem - Unable to access customer encryption key

When trying to create, edit, or test a relational connection, you may receive the following error message:

```
400 - Encryption Key Error. Please Contact Administrator:
Unable to access customer encryption key.
```

You are unable to access the relational source.

Solution

The encryption keyfile is missing from the Trifacta® deployment, or the keyfile has been moved without updating the platform of the new location.

You must create and deploy this keyfile, which is required for ensuring that encrypted usernames and passwords are used in relational connections.

NOTE: This keyfile must be created and deployed before any relational connections are created. Deployment requires access to the file system on the Trifacta node.

After you have deployed the keyfile, you must configure the platform to point to its location. A platform restart is not required.

For more information, see *Relational Access*.

Problem - Retrieving sample data for large relational tables is very slow

In some cases, you may experience slow performance in reading from database tables, or previews of large imported datasets are timing out.

Solution

In these cases, you can experiment with the number of records that are imported per database read. By default, this value is 25000.

To improve performance, you can modify the following setting.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

```
"data-service.sqlOptions.limitedReadStreamRecords": 25000,
```

To improve performance, you can try lowering this limit incrementally. Avoid raising this limit over 100000, which can overwhelm the browser.

Supported Connection Credential Types

Contents:

- *API Key*
- *API Key with Token*
- *Azure Token SSO*
- *AWS*
- *AWS Key/Secret*
- *Basic*
- *Basic app*
- *Basic with app token*
- *conf*
- *HTTP Header-Based Authentication*
- *HTTP Query-Based Authentication*
- *IAM DB User*
- *IAM Role Arn*
- *Kerberos Delegate*
- *Kerberos Impersonation*
- *Key/Secret*
- *No Authentication*
- *OAuth 2.0*
- *Password*
- *Security Token*
- *SSH Key*
- *SSH Tunneling Basic*
- *SSH Tunneling SSH Key*
- *Transaction Key*
- *User with API Token*
- *Reference Information*
 - *Connections by Credential Type*
 - *API References*

This section contains general reference information on the credential types that are supported for use in connections from the Trifacta® application. A **credential type** defines the authentication or account information that must be provided to the authenticating application.

NOTE: Some credential types may not be available in your product edition.

API Key

This credential type requires generation of an API key within the target application. This key must be inserted as part of the connection definition in the Trifacta application.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["apiKey"]
```

API Key with Token

This credential type requires an API key generated by the target application, as well as an access token tied to the API key.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["apiKeyWithToken"]
```

Azure Token SSO

Connect to Azure-hosted resources using the Azure Single Sign On (SSO) token for the authenticating user.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["azureTokenSso"]
```

AWS

AWS-specific credentials. Used for Redshift connections.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["aws"]
```

AWS Key/Secret

These AWS-specific credentials use a key/secret combination to authenticate to AWS systems, such as Amazon Dynamo DB and Amazon Athena.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["awsKeySecret"]
```

Basic

A simple username/password can be provided to the authenticating application.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["basic"]
```

Basic app

The basic app credential type requires that a private app be created in the target application. Access through this app needs an AppId and Password combination.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["basicApp"]
```

Basic with app token

This basic authentication mechanism requires three pieces of information: Username , Password and Application Token . All of these are available in through the target application.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["basicWithAppToken"]
```

conf

For this credential type, the connection credentials are stored in `trifacta-conf.json`, a JSON configuration file stored on the node hosting the product.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["conf"]
```

HTTP Header-Based Authentication

Used for REST API connections, these credentials are submitted as key/value pairs in the HTTP request.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["httpHeaderBasedAuth"]
```

HTTP Query-Based Authentication

Used for REST API connections, these credentials are submitted as key/value pairs in URL query parameters.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["httpQueryBasedAuth"]
```

IAM DB User

This credential type leverages an IAM role to access Amazon Redshift databases. The IAM role must be specified as part of the connection definition.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["iamDbUser"]
```

IAM Role Arn

This credential type uses an IAM role to access external S3 buckets, which are not defined as part of the base storage layer.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["iamRoleArn"]
```

Kerberos Delegate

Connection uses the Kerberos-delegated principal to connect to a relational database. No credentials are submitted as part of the connection definition. This method requires additional configuration.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["kerberosDelegate"]
```

Kerberos Impersonation

Connection uses the Kerberos impersonation principal for the user to connect to the database. No credentials are submitted as part of the connection definition.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["kerberosImpersonation"]
```

Key/Secret

When accessing an external S3 bucket, you can apply key-secret combinations as part of your connection definition. This authentication mechanism consists of an AWS Access Key ID and an AWS Access Secret ID.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["keySecret"]
```

No Authentication

Some connection types do not require credentials to be submitted to them.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["noAuth"]
```

OAuth 2.0

OAuth 2.0 credentials can be used to connect a client in the Trifacta application to the client app created in the target system.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["oauth2"]
```

NOTE: Additional configuration may be required to enable this credential type for a specific connection type.

Password

A single password value is required for authentication.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["password"]
```

Security Token

This credential type requires the insertion of a single security token as part of the connection definition. This security token must be generated from the targeted application.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["securityToken"]
```

SSH Key

Used for SFTP connections, this credential type requires that you insert an SSH key generated from the host server of the FTP site.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["sshKey"]
```

SSH Tunneling Basic

For SSH tunneling connectivity, you can use a simple username and password set of credentials. This credential type can be applied to various connection types.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["sshTunnelingBasic"]
```

SSH Tunneling SSH Key

For SSH tunneling connectivity, you can use a username and SSH key as a set of credentials. This credential type can be applied to various connection types.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["sshTunnelingSshKey"]
```

Transaction Key

This credential type uses a Login ID and Transaction Key to authenticate.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": ["transactionKey"]
```


User with API Token

This credential type requires a user identifier and an API token associated with that user to authenticate to the server.

Trifacta API attribute:

When creating a connection via API, the following attribute and value must be inserted as part of the connection definition:

```
"credentialType": [ "userWithApiToken" ]
```

Reference Information

Connections by Credential Type

Credential Type	Connection Type
apiKey	Airtable, Freshdesk, HubSpot, Mailchimp
apiKeyWithToken	Trello
awsKeySecret	Amazon Athena, Amazon DynamoDB
azureTokenSource	Azure SQL Database
basic	MariaDB on Amazon RDS, MySQL on Amazon RDS, Oracle DB on Amazon RDS, PostgreSQL on Amazon RDS, SQL Server on Amazon RDS, Apache Impala, Azure SQL Database, Cassandra DB, MySQL on Google Cloud SQL, PostgreSQL on Google Cloud SQL, SQL Server on Google Cloud SQL, Cockroach DB, DB2, Greenplum, IBM DB2, REST API, Jira by Atlassian, Magento, MariaDb, MongoDB, MongoDB Atlas, MySQL, Oracle Database, PostgreSQL, Presto, ServiceNow, SFTP, SharePoint, Snowflake, Splunk, Azure Synapse Analytics (Formerly Microsoft SQL DW), Microsoft SQL Server, Tableau Server, Teradata, Trino, Zendesk
basicApp	Shopify
basicWithAppToken	Quickbase
conf	Databricks, Amazon Glue, Hive
httpHeaderBasedAuth	\$strConnectionType
iamDbUser	Amazon Redshift
iamRoleArn	Amazon Redshift
keySecret	External Amazon S3
noAuth	REST API, Presto, Trino
oauth2	Asana, Microsoft Advertising, Microsoft Dynamics 365 Sales, Exact Online, Facebook Ads, Google Ads, Google Analytics, Google Data Catalog, Google Spanner, Google Sheets, Instagram Ads, LinkedIn Ads, Marketo, NetSuite, Pinterest, QuickBooks Online, Salesforce, Smartsheet, Snowflake, SurveyMonkey, Xero, YouTube Analytics
password	Redis
securityToken	SalesForce (Deprecated), SalesForce
sshKey	SFTP
transactionKey	Authorize.Net

API References

In the request and response for actual connections, the attribute `credentialTypes` is used as a String value:

```
{
  "id": 37,
  "host": "postgres.example.com",
  "port": 5432,
  "vendor": "postgres",
  "params": {
    "connectStrOpts": "",
    "database": "mydb"
  },
  "ssl": false,
  "vendorName": "postgres",
  "name": "Postgres20200417182437287",
  "description": "",
  "type": "jdbc",
  "isGlobal": false,
  "credentialType": "basic",
  "credentialsShared": false,
  "uuid": "myUniqueId",
  "disableTypeInference": false,
  "createdAt": "2020-04-17T18:25:04.518Z",
  "updatedAt": "2020-04-17T18:25:04.530Z",
  ...
}
```



Copyright © 2022 - Trifacta, Inc.
All rights reserved.