TIBCO® Data Virtualization Adapter Guide

Version 8.0

Last Updated: *November 30, 2018*

Two-Second Advantage®

Important Information

Contents

Preface

For information on the following, see the *TDV User Guide*:

- Adding a Data Source
- Introspecting a Data Source
- Testing the Connection to Your Data Source

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, please visit:

- https://docs.tibco.com

Product-Specific Documentation

The following documents form the TIBCO® Data Virtualization(TDV) documentation set:

- *TIBCO TDV and Business Directory Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.
- TDV Installation and Upgrade Guide
- TDV Administration Guide
- TDV Reference Guide
- TDV User Guide
- TDV Security Features Guide
- Business Directory Guide
- TDV Application Programming Interface Guide
- TDV Tutorial Guide
- TDV Extensibility Guide
- TDV Getting Started Guide
- TDV Client Interfaces Guide
- TDV Adapter Guide
- TDV Discovery Guide

- TDV Active Cluster Guide
- TDV Monitor Guide
- TDV Northbay Example

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website mainly in the HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

Documentation for TIBCO Data Virtualization is available on https://docs.tibco.com/products/tibco-data-virtualization-server.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit https://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to https://community.tibco.com.

Document Change History

The table below provides the revision history for this Guide.

Version Number	Issue Date	Status	Reason for Change
8.0	November 2017		New version. Updates to several adapters and 3 new adapters added.

Configuring Adapters

This topic describes the configuration of optional TDV adapters. These topics are covered in this topic:

Adapter Data Type Limitation

There are several data source types that have two or more adapters that can be used to interpret the data coming into TDV. IF you have designed some views based on objects imported with one adapter and you switch adapter types, say Salesfore for Salesforce SSO, your introspected data might be assigned to different data types. For example, data that was previously introspected as DOUBLE might now be assigned to the DECIMAL data type.

Dynamics CRM Adapter ID Ordering

Id's in Dynamics CRM have special ordering and when you do ORDER BY ASC or DESC based on the ID, the data might not order the way you expect until you get used to the way Dynamics CRM organizes itself.

Upgrading the ADO.NET Driver for Sharepoint Adapters

To upgrade the ADO.NET driver for Sharepoint Adapters

1. Read any README files included with or associated with the download file.
2. Locate and extract the drivers zip file.
3. Make sure there is no application using or the holding ADO.NET driver.
4. Uninstall the old ADO.NET driver.
5. Install the new ADO.NET driver.
6. Update connection settings in the client and development tools.
7. Refer to the TDV Client Interfaces Guide for details on complete configuration of the ADO.NET driver.
8. Restart any application that's using the ADO.NET driver.]

Configuring OAuth 2.0 for TDV Advanced Adapters

If your Advanced Data Source Adapters will make use of OAuth authentication, then there is extra configuration that you must perform. The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service for:

- A resource owner by authorizing the resource owner and the HTTP service
- A third-party application to manage authorization separately

This section includes:

OAuth Options Provided by TDV Advanced Adapters

TDV provides the following OAuth options:

Initiate OAuth field options	Description
GETANDREFRESH	Typically, this option can be established quickly by most users. The Get and Refresh option uses the client id and client secret to initiate the OAuth as described in the OAuth 2 RFC and requires interaction between client, resource owner, and the Authorization and Resource Servers. A browser must be launched on the local machine to enable the granting of permissions. The OAuth settings file must point to a location on the TDV Server machine. Required field: Client Id, Client Secret, OAuth Settings file Location
OFF	This configuration uses an OAuth Access Token to authorize to the back end; it doesn't initiate any OAuth flow. The OAuth access token needs to be provided as part of configuration. This method is recommended if the access token is already obtained using other means. Most OAuth providers have a way to generate access tokens, with the help of developer consoles or APIs. Access tokens do expire and it is your responsibility to provide a new access token, when the previous one expires. This makes it suitable for one time use or when the access tokens have long life. Required Field: OAuth Access Token
REFRESH	This option refreshes the access token using a refresh token, client id, and client secret, when the current access token is expired and the refresh is transparent to the user. It stores the new access token in the OAuth settings file. Typically, users do not need to enter anything. This option is best where the Studio and the TDV Server cannot be run on the same machine. It does require that a refresh token be obtained out side of TDV using OAuth provider tools. Required fields: OAuth Refresh Token, Client Id, Client Secret, OAuth Settings file Location

Preparing Eloqua for OAuth Access through TDV

The Eloqua data source adapter does not permit the use of the loopback address "localhost" or "127.0.0.1" in the callback URL. To use Eloqua with TDV and OAuth, you must configure your system resolver to resolve the new hostname you choose for this purpose to the loopback address 127.0.0.1, and use that new hostname in the callback URL. You must also identify an unused port.

To prepare Eloqua for OAuth access

1. On your TDV Server host, add a new hosts file entry that maps the loopback address 127.0.0.1 to a fully qualified hostname not actually used on your network, such as "eloqualoopback.mycisserver.com."

   For example, on Windows add the following entry to your c:\windows\system32\drivers\etc\hosts file:

```
127.0.0.1 eloqualoopback.mycisserver.com  # the fully qualified
hostname for the Eloqua Callback URL
```

2. Choose an available port on the host running TDV for use in the Callback URL. For example: 12481.
3. Log into Eloqua and configure your app's AppCloud Developer Settings to reference that hostname and port in the Callback URL. For example:

```
https://eloqualoopback.mycisserver.com:12481/
```

   The https protocol is required.

Configuring GETANDREFRESH OAuth access for Advanced Data Source Adapters

The GETANDREFRESH option is the most typical configuration method for configuring OAuth access for your TDV Advanced Data Source adapters.

To configure GETANDREFRESH OAuth access for Advanced Data Source Adapters

1. Start the TDV from the UNIX or Windows command line.
2. Start Studio on the machine where you are running the TDV Server.
3. Add a new data source for one of your Advanced Data Source Adapters.
4. Provide a unique name for your data source.
5. On the Basic tab, type your company name (or account identifier).

6. On the Advanced tab, enter values for the following fields:

Field	Description of Value
Initiate OAuth	Typically, et this to GETANDREFRESH. Or a refresh token can be obtained from the OAuth provider and provided along with the client id and client secret. Getting a Refresh token requires experience in using the developer APIs and console of the OAuth provider. Also, the token has to be obtained using the same client id and client secret that is configured in the data source.
OAuth Client Id App ID	Set this to the client Id (app Id) of your data source.
OAuth Client Secret App Secret	Set this to the client secret of your data source.
Other	Set this to the value of your data sources callback URL. For Eloqua, obtaining the callback URL requires some extra steps. `https://eloqualoopback.mycisserver.com:12481/`

After the initial authentication is run and the tokens have been obtained, they are written into the configured OAuth settings file.

7. Save your new data source.
8. From the Studio resource tree, open the data source that you just added, and select Test Connection. Or introspect the data source.

   The adapter opens the OAuth endpoint in your system default browser.
9. Using the browser, log in and grant TDV permission to the adapter application.

   For example, Eloqua calls your specified callback URL, appending the access token and other needed values, such as:

```
https://eloqualoopback.mycisserver.com:12481/callback#access_token
=2YotnFZFEjr1zCsicMWpAA&token_type=bearer&expires_in=86400&state=x
yz
```

10. If or when the authentication certificates expire, you must perform this procedure again to allow TDV permission to the data in the data source.

Replicating your OAuth Configuration on Other Machines

After initial testing and development work with your Advanced Data Source Adapters within TDV and Studio, it is typical that authorization settings will need to be migrated or replicated on other machines within your TDV environment. You can do this using CAR files and by migrating the OAuth settings file.

Occasionally, your usage focus might also require the replication of the authorization settings. Some of these usage focuses include:

Usage Focus	Description
Importing/Exporting Archives and Deployment Manager	When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuth settings file is not automatically imported or migrated. OAuth settings file needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.
Clustered Environment	In a cluster, data source configuration is synced across the members of the cluster. For OAuth providers, who allow multiple valid tokens, configure OAuth settings location to be same path. If that is not possible, provide a path on the shared file system, which is accessible to all the members. Shared file system path is needed in case, where the OAuth provider does not support multiple valid tokens.

To replicate your OAuth configuration on Other Machines

1. From the Studio resource tree open the data source for which you want to replicate OAuth configuration.
2. On the Advanced tab, locate and save the value of the following field:

OAuth Settings Location	The value of this field is the location where the adapter writes the authentication information. If working in a clustered environment, make sure this location is central to all the nodes of the cluster, or replicate the OAuth settings file to each cluster

3. Create a CAR file export of the data source for which you want to replicate OAuth configuration.
4. Copy the OAuth settings file and the CAR file to the new host machine.
5. Copy the settings file to the same directory location as was specified in the OAuth Settings Location for the data source that you are replicating.

Doing this allows a data source imported from the source to be used in target without modifications.

6. Use Studio on the new host machine to import the CAR file of the data source you are migrating.
7. Test the connection of the data source. If necessary resolve any issues.

Installing the Siebel Data Bean JARs

For the Siebel adapter, you must install the Siebel Data Bean JARs to enable connectivity to Siebel. Because these JARs are present on the Siebel Server itself, you might need assistance from your Siebel System Administrator. Follow the process below.

To install Siebel Data Bean JARs

1. Create a directory for the Siebel Data Bean JARs in the following location; <version> is the Siebel version:

   <TDV_install_dir>/apps/dlm/app_ds_siebel/lib/<version>

2. Copy Siebel.jar and SiebelJI_enu.jar from directory below to the directory created in the previous step. <Siebel> is the root directory of your Siebel Server.

   <Siebel>/eaiconn/CLASSES

Registering with the SAP System Landscape Directory

After installing the TIBCO software, you must register the installation with the SAP System Landscape Directory if you are using this feature. The procedures for both Windows and UNIX are shown below.

Registration Procedure for Windows

To register the installation with the SAP System Landscape Directory on Windows

1. Run the script sap_sld_register_util.bat under <TDV_install_dir>/bin with command-line arguments <SLDHost> <Port> <Username> <Password>.
2. If connecting through a proxy, modify the script line set JAVA_OPTS= to:

```
set JAVA_OPTS=-Dhttp.proxyHost=<server> -Dhttp.proxyPort=<port>
```

Registration Procedure for UNIX

To register the installation with the SAP System Landscape Directory on UNIX

1. Run the script sap_sld_register_util.sh under <TDV_install_dir>/bin with command-line arguments <SLDHost> <Port> <Username> <Password>.
2. If connecting through proxy, modify the script line JAVA_OPTS= to:

```
JAVA_OPTS=-Dhttp.proxyHost=<server> -Dhttp.proxyPort=<port>
```

Installing the SAP Java Connector Library

You must install the SAP Java Connector Library (SAP JCo), version 3.0.9, on the same computer as TDV. SAP JCo is available through the SAP Service Marketplace at www.service.sap.com. You need to be a registered customer to download from there. After you sign in, navigate to SAP Java Connector > Tools & Services and download the JCo distribution for the platform running TDV, then refer to the next section to validate the installation.

Note: In SAP JCo version 2.1.5, passwords are no longer automatically converted to uppercase when they are sent to SAP; instead, case is preserved, which could lead to log on failure. The easiest way to address this is to change the data source to use an uppercase version of passwords. See SAP Notes 817880 and 792850 for more information.

It is best not to install the SAP Adapter on the same machine as the SAP GUI, to avoid a possible conflict between JCo versions.

- SAP JCo and Large Volumes of Data, page 29
- Installing SAP JCo on Windows, page 29
- Testing the SAP JCo Installation on Windows, page 29
- Installing SAP JCo on UNIX, page 30
- Testing the SAP JCo Installation on UNIX, page 31

SAP JCo and Large Volumes of Data

JCo has known memory limitations when processing queries. Severe cases can affect the main TDV Java process. In such cases, we recommend that large queries be divided into small queries.

Installing SAP JCo on Windows

You need to obtain the SAP Java Connector for your machine from SAP. TIBCO has tested version 3.0.9 of the SAP Java Connector.

To install SAP JCo on Windows

1. Unzip the SAP JCo zip file into a temporary directory.
2. Copy sapjco3.jar to <TDV_install_dir>\jre\lib\ext.
3. Copy sapjco3.dll to one of these directories:
   - For Windows 32: <TDV_install_dir>\apps\common\lib
   - For Windows 64: <TDV_install_dir>\apps\common\lib\win64
4. Add a system PATH variable to point to the directory where you put your library files in the previous step.
5. Verify that the Windows system directory includes the files MSVCR71.DLL and MSVCP71.DLL.

   These DLLs are the Shared C Runtime (CRT) Components required by SAP JCo but are not shipped with it. If these DLLs are missing, install the Microsoft .NET Framework SDK Version 1.1. Microsoft .NET Framework SDK is available from the Microsoft Developer Network (MSDN) download site.

Testing the SAP JCo Installation on Windows

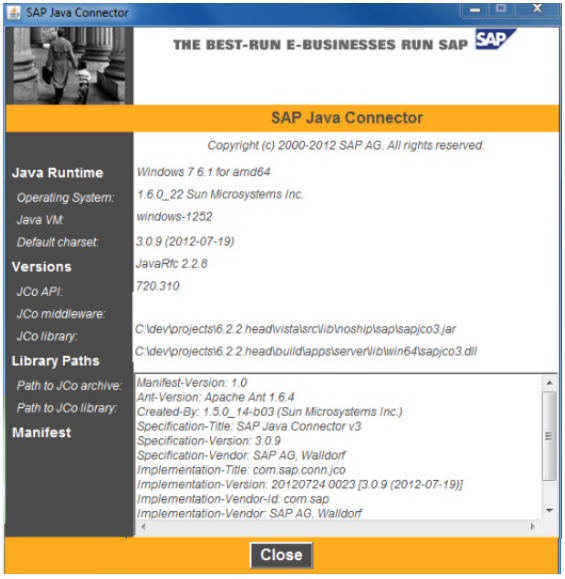
Be sure to test that the SAP JCo installation works correctly before using TDV.

To test the installation of SAP JCo on Windows

1. Run **the sapjco3.jar Java** executable using one of these commands:

Platform	Command
32-bit	`<TDV_install_dir>\jre\bin\java -Djava.library.path="<TDV_install_dir>\apps\common\lib" -jar "<TDV_install_dir>\jre\lib\ext\sapjco3.jar"`
64-bit	`<TDV_install_dir>/jre/bin/java -Djava.library.path="<TDV_install_dir>\apps\common\lib\win64" -jar "<TDV_install_dir>\jre\lib\ext\sapjco3.jar"`

2. Verify that running sapjco3.jar results in a screen that looks like this.



If an error message appears instead, SAP JCo needs to be re-installed.

Installing SAP JCo on UNIX

Follow the instructions to install SAP JCo based on the type of UNIX system you have. You need to obtain the SAP Java Connector for your machine from SAP. TIBCO has tested versions of the SAP Java Connectors up to 3.0.9.

To install SAP JCo (Linux, Solaris, and AIX):

1. Unzip the SAP JCo tgz file for your machine into a temporary directory.
2. Copy sapjco3.jar to <TDV_install_dir>/jre/lib/ext directory, where <TDV_install_dir> is the root directory of your TDV Server.
3. Copy libsapjco3.so into one of these directories:
   - Linux 32: <TDV_install_dir>/jre/lib/i386
   - Linux 64: <TDV_install_dir>/jre/lib/amd64
   - Solaris 64: <TDV_install_dir>/jre/lib/sparcv9
   - AIX 64: <TDV_install_dir>/jre/lib/ppc64
4. (For Linux and Solaris) Add a system LD_LIBRARY_PATH variable to point to the directory where you put your library files in the previous step.
5. (For AIX) Add a system LIBPATH variable to point to the directory where you put your library files in the previous step.

To install SAP JCo (HP-UX):

1. Unzip the SAP JCo tgz file for your machine into a temporary directory.
2. Copy sapjco3.jar to <TDV_install_dir>/jre/lib/ext directory, where <TDV_install_dir> is the root directory of your TDV Server.
3. Copy libsapjco3.sl into this directory:
   - HP-UX 64: <TDV_install_dir>/jre/lib/PA_RISC2.0W
4. Add a system SHLIB_PATH variable to point to the directory where you put your library files in the previous step.

Testing the SAP JCo Installation on UNIX

It is critical to verify that SAP JCo is working before you attempt to connect to SAP using TDV.

To test the UNIX installation

1. From the command line, run the following command:

   <TDV_install_dir>/jre/bin/java -Djava.library.path=/<LIBRARY_PATH> –jar <TDV_install_dir>/jre/lib/ext/sapjco3.jar –stdout

Where LIBRARY_PATH is the path for your system:

- Linux and Solaris: $LD_LIBRARY_PATH
- AIX: $SHLIB_PATH
- HPUX: $LIBPATH

2. Verify that your output resembles the text below. If this command runs without error and the JCo API, middleware, and libraries have version information listed, then SAP JCo is properly installed.

```
--------------------------------------------------------------------
------
|                              SAP Java Connector
|
|        Copyright (c) 2000-2012 SAP AG. All rights reserved.
|
|                             Version Information
|
--------------------------------------------------------------------
------
Java Runtime:
 Operating System:       AIX 5.3 for ppc64
 Java VM:                1.6.0 IBM Corporation
 Default charset:        ISO-8859-1
Versions:
 JCo API:                3.0.9 (2012-07-19)
 JCo middleware:         JavaRfc 2.2.8
 JCo library:            720.310
Library Paths:
 Path to JCo archive:    /opt/TDV/jre/lib/ext/sapjco3.jar
 Path to JCo library:    System-defined path to libsapjco3.a
--------------------------------------------------------------------
------
|                                  Manifest
|
--------------------------------------------------------------------
------
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.6.4
Created-By: 1.5.0_14-b03 (Sun Microsystems Inc.)
Specification-Title: SAP Java Connector v3
Specification-Version: 3.0.9
Specification-Vendor: SAP AG, Walldorf
Implementation-Title: com.sap.conn.jco
Implementation-Version: 20120724 0023 [3.0.9 (2012-07-19)]
Implementation-Vendor-Id: com.sap
Implementation-Vendor: SAP AG, Walldorf
Main-Class: com.sap.conn.jco.rt.About
--------------------------------------------------------------------
------
```

Verifying a Successful Activation of an Adapter

Verification ensures that the adapter (which includes Advanced Data Source Adapters) was successfully installed and available as a data source.

To verify a successful activation

1. Start and log into Studio.
2. In the resource tree on the left, select a location to add a data source, right-click, and select New Data Source.

   In the New Physical Data Source window that opens, your adapter should appear in the list of data source drivers, indicating a successful installation.
3. Click Cancel.

Increasing the Maximum Size of Excel Sheets for SharePoint

Getting data from an Excel worksheet that exceeds the maximum allowable size in SharePoint, results in a "413 Request Entity Too Large" message.

To avoid the message, you must increase the size of workbook allowed to be opened by Excel Services ands the default IIS request size.

To increase the size of a workbook in Excel Services

1. Open SharePoint Central Administration.
2. Navigate to Application Management > Manage Service Applications > ExcelServiceApp > Trusted File Locations.
3. Select the appropriate Address.
4. Under the Workbook Properties section, locate a Maximum Workbook Size option.
5. Set this value to the required file size in MB.
6. Click OK to save the setting.

   To avoid error messages, make sure the value is higher than the largest workbook size in SharePoint.

To increase the default IIS request size

1. Open a command window.
2. Navigate to

```
<drive>\Windows\System32\inetsrv
```

3. Type the following command:

```
appcmd.exe set config "http://localhost/<rmft_virtual_dir>"
-section:system.webServer/serverRuntime
/uploadReadAheadSize:"request size in bytes" /commit:apphost
```

For example:

```
appcmd.exe set config
"http://localhost/repliweb-mft"-section:system.webServer/serverRun
time /uploadReadAheadSize:"100000000" /commit:apphost
```

4. You can check the configuration using the following URL in your browser:

```
https://<your site>.com/_vti_bin/ExcelRest.aspx/<document
library>/<folder (optional)>/<file name>.xlsx/OData/<sheet name>
```

Using the Oracle E-Business Suite Adapter

This topic describes:

About Oracle E-Business Suite Data Source

For installation requirements, see the *TDV Installation and Upgrade Guide*.

Oracle EBS modules are classified into three large groups: Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Supply-Chain Management (SCM). This adapter supports the following applications

- CRM
- Financials
- Human Resources
- Manufacturing
- Supply Chain
- Items
- Purchasing
- Requisitions
- Sales Orders
- Transactions

Select Oracle E-Business Suite on 10g.

The 9i and 10g are historical labels for the database version number and have not been updated in the Studio UI; choose the one closest to the target database version and click Next to continue.

Oracle EBS Basic Tab

For Connection Information on the Basic tab, supply the required information in the following input fields:

Field	Description
Host	Name of the machine hosting Oracle E-Business Suite or the host machine's IP address.
Port	Port number of Oracle database instance. Default is 1521.
Service Name	Database ID for the Oracle E-Business Suite back-end database.
Database User Name and Database Password	User name and password to access Oracle instance. The user name "apps" is the default for Oracle E-Business Suite.Database User Name and Database Password are different from Application User Name and Application Password. Database means the credentials are used to connect to the Oracle database instance. Application credentials are used for authenticating as a user of Oracle E-Business Suite.
Transaction Isolation	Choose between Read Committed and Serializable. Read Committed can prevent dirty reads; Serializable can prevent dirty reads, non-repeatable reads and phantom reads.
Authentication	Choose between BASIC and KERBEROS i
Oracle E-Business Suite Version	Choose the right version or the nearest version. It is used when Multi-Org Enabled View is invoked. A different directory for a specific Oracle E-Business Suite version may have a different view definition. • 11.5.8—select for EBS version 11.5.8 • 11.5.10—select for EBS version 11.5.10 • 12.1.3—select for EBS version 12.1.3

Field	Description
Row Level Security Filter	Choose one of the following: • None—No row level filter is applied • Retrieve data by setting connection context —No row level filter is applied. Data are retrieved in an Oracle E-Business Suite context. • Retrieve data based on user responsibilities —Enforces row-level authorization checking for tables supporting this feature. Users are able to see only those rows belonging to organizations they have rights to access, as specified by their user (Application) credentials. — Pass-through Login—Works in combination with the Application User Name and Application Password options. — Disabled (default)—Non-pass-through mode. — Enabled—Pass-through mode; the pass-through credential will replace the Application credential when connecting to Oracle E-Business Suite. — Application User Name and Application Password — User's credential to access Oracle E-Business Suite. In Pass-through mode these values are replaced with pass-through credentials provided by the user. — Multi-Org Reporting—Specify whether to use the Multiple Organization View (default) or the Oracle EBS Single Organization View. • Retrieve data by fixed organization IDs—Enforces row-level authorization checking for tables supporting this feature. Users are able to see only those rows belonging to organizations specified by a comma-separated organization ID list.

Oracle EBS Advanced Tab

Some properties refer directly to the configuration of the Oracle E-Business Suite Server and must be provided by an Oracle E-Business Suite Administrator. Other properties are specific to TDV and how it interacts with Oracle E-Business Suite.

Connection Pool Minimum Size, Maximum Size, Idle Timeout(s), and Maximum Connection Lifetime—These control the JDBC connections made to Oracle E-Business Suite, specifying timeout in seconds, the minimum and

maximum number of simultaneous connections in a Oracle E-Business Suite connection pool.

Connection Validation Query—A simple query to test whether the connection is valid or not. You can leave it empty, or use simple queries like “select * from dual.”

Organization Id Cache Timeout Seconds—Length of time to cache user security information. After this time the security information is refreshed from Oracle E-Business Suite. If an Oracle E-Business Suite user's security information is changed before the cache is refreshed, the changes will not be available to the Oracle E-Business Suite data source until the cache is refreshed.

Security for Oracle E-Business Suite with TDV

This topic describes TDV’s security support for Oracle E-Business Suite. Security involves two parts: authentication and access control.

The following topics are covered:

Understanding Authentication for Oracle E-Business Suite

The data source properties Database Username and Database Password are needed to connect to the Oracle database. This connection is used to verify application credentials, gather access information, execute queries, and read query results.

If Enable Multi-Organization is checked, data in Oracle E-Business Suite is restricted based on the user's application credentials. Application credentials are specified in the data source properties Application Username and Application Password, or by enabling pass-through login. The application credentials are authenticated using Oracle E-Business Suite’s API and used to obtain the users permitted organizations and responsibilities.

Understanding Access Control for Oracle E-Business Suite

An application user has one or more responsibilities. A responsibility is a level of authority in Oracle E-Business Suite that lets users access only those Oracle E-Business Suite functions and data appropriate to their roles in an organization. For data access control, each responsibility allows access to a set of books, such as U.S. Operations or German Sales, or an organization, such as New York Manufacturing or New York Distribution.

Each responsibility has one organization (org_id) associated with it, which can be set at system, site, responsibility, or user level. The organization limits users to data relevant to their organization. For example, you can limit the access of order administration clerks to sales orders associated exclusively with their sales office.

Information limited by organization is stored in Multiple Organization-enabled tables. Their names typically end in _ALL or _ALL_B. When displayed during introspection, these tables are notated with the word Secure after their names. When querying against these tables with Enable Multi-Organization checked, only information visible to the current user is retrieved. The filtering is transparent to the user.

You can specify which inventory organization is available to users by responsibility. A single responsibility, by system level or responsibility level, determines an operating unit. So a user with multiple responsibilities may have multiple organizations visible. The union of all the user's visible organizations determines how information is filtered in their queries.

- Reporting with Multiple Organizations, page 39
- Using Organization Names in Queries , page 40

Reporting with Multiple Organizations

For reporting purposes users may need to read information for which they lack the authority, as defined by their responsibilities. To achieve the flexibility required in reporting, the Oracle Applications profile option, “MO: Top Reporting Level” can be used to expand the list of organizations visible to users.

The value of this profile option is set to “Operating Unit” at the site level. Your Oracle E-Business Suite system administrator can set this profile option at the responsibility level. The following table shows the access given to a user depending on the setting of the Top Reporting Level profile option.

Top Reporting Level Setting	Enables Users to
Set of Books	View all the data belonging to the set of books, legal entities, and operating units.

Top Reporting Level Setting	Enables Users to
Legal Entity	View all the data belonging to the legal entity and operating units.
Operating Unit	View data only in the operating unit assigned to their responsibility.

Using Organization Names in Queries

All secure tables (enabled for multiple organization use) contain a column named Organization ID. This query translates the organization id into meaningful organization name:

```
SELECT name
FROM /shared/DataService_Oracle_EBS11i_on_9i/"Data
Services"/"Human Resources"/Organizations Organizations
WHERE "Organization Id" = 204
```

An application user has one or more responsibilities. A responsibility is a level of authority in Oracle E-Business Suite that lets users access only those Oracle E-Business Suite functions and data appropriate to their roles in an organization. For data access control, each responsibility allows access to a set of books, such as U.S. Operations or German Sales, or an organization, such as New York Manufacturing or New York Distribution. For more details, please refer to "Multiple Organizations in Oracle Applications, Release 11i."

Each responsibility has one organization (org_id) associated with it, which can be set at system, site, responsibility, or user level. The organization limits users to data relevant to their organization. For example, you can limit access for order administration clerks to sales orders associated exclusively with their sales office.

Information limited by organization is stored in Multiple Organization-enabled tables. Their names are typically ended in _ALL or _ALL_B. When displayed during introspection, these tables are notated with word "Secure" after their names. When querying against these tables with Enable Multi-Organization checked, only information visible to the current user is retrieved. The filtering happens transparently to the user.

You can specify which inventory organization is available to users by responsibility. A single responsibility, by system level or responsibility level, determines an operating unit. So a user with multiple responsibilities may have multiple organizations visible. The union of all the user's visible organizations determines how information is filtered in their queries.

Using Data Filters with Oracle E-Business Suite

With TDV's query engine, you can dynamically apply a data filter to control the data retrieved. So when the situation changes—for instance, a report for one organization needs to extend to several organizations—you just need to change the filter in the data source configuration.

This filter mechanism works well with Oracle's multi-org reporting. In Oracle's own database view, many views are limited to one organization. There is no simple way to query on these views for multiple organizations or all the organizations. With the organization ID filter and a simple view translation, multiple organizations' data can be retrieved from the same view. For more information about Oracle's multi-org reporting, see Understanding Multi-Org Enabled Views, page 43.

For example, the following table shows the results returned for the same query "SELECT * FROM AP_CHECKS_V":

	Query without Multi-Org Enabled View	Query with Multi-Org Enabled View
All Rows of Data	Not Available	Available
One Organization's Data	Available	Available
Multiple Organization's Data	Not Available	Available

Use the following topics to learn more:

- Choosing a Data Filter, page 41
- List of Shipped Multi-Org Enabled Views, page 42
- How Credentials Are Used, page 43

Choosing a Data Filter

Different data filters return different stripes of data from the same view. The following table shows the results returned for the same query "SELECT * FROM AP_CHECKS_V":

	SELECT * FROM AP_CHECKS_ALL	Query without Multi-Org Enabled View	Query with Multi-Org Enabled View
None	All rows	No rows	All rows

	SELECT * FROM AP_CHECKS_ALL	Query without Multi-Org Enabled View	Query with Multi-Org Enabled View
Org ID(204) in connection context	All rows	Rows for org 204	Rows for org 204
Pass-through or fixed application username and password	Rows for the org the specific user has access to	No rows	Rows for the org the specific user has access to
List of organization IDs	Rows for the list of org ids	No rows	Rows for the list of org ids

For views that are not related to organization ID, all rows are returned in all cases.

List of Shipped Multi-Org Enabled Views

The following views are translated and extended to cover multiple organizations. The view definitions reside in the installation directory depending on the Oracle E-Business Suite version you have chosen.

C:\<TDV_install_dir>\apps\dlm\app_ds_oa\conf\<ver_num>

```
AP_BANK_ACCOUNT_USES_V
AP_CHECKS_V
AP_HOLDS_OVERVIEW_V
AP_INVOICES_V
AP_INVOICE_PAYMENTS_V
AP_PAYMENT_SCHEDULES_V
AP_VENDORS_V
AP_VENDOR_SITES_V
AR_ADJUSTMENTS_V
AR_CASH_RECEIPTS_V
AR_CUSTOMER_CALLS_V
AR_MEMO_LINES_VL
AR_RECEIVABLE_APPLICATIONS_V
OE_ORDER_HEADERS_V
OE_ORDER_LINES_V
PER_ASSIGNMENTS_V
PER_PEOPLE
PER_PEOPLE_V
PO_DOCUMENT_TYPES_VL
PO_HEADERS_V
PO_LINES_V
PO_REQUISITION_HEADERS_V
PO_REQUISITION_LINES_V
RA_CUSTOMER_TRX_LINES_V
RA_CUSTOMER_TRX_PARTIAL_V
RA_CUST_TRX_LINE_GL_DIST_V
```

How Credentials Are Used

The following table describes various credentials and their usage:

Credentials	Usage
Database User Name / Password	To establish database connections
Application User Name / Password	Authentication, Access Control
Pass-Through User Name / Password	Overwrite Application User Name / Password for Authentication, Access Control

Understanding Multi-Org Enabled Views

Multi-Org Enabled Views are one way to use the Oracle E-Business Suite Adapter's dynamic filtering. The views in the existing Oracle E-Business Suite system are limiting. For example, view AP_CHECKS_V has some useful information, but for only one organization. But with the Oracle E-Business Suite Adapter, you can select rows of all organizations, or a specific list of organizations, from AP_CHECKS_V. This is done by replacing limitations in the view.

For example,

```
WHERE org_id = 204
```

You can replace this with

```
WHERE org_id in (204, 600)
```

See List of Shipped Multi-Org Enabled Views, page 42" to locate the translated views' definition.

When TDV is querying the Oracle database, if a view's definition is found under the directory, the view's definition will be used.

How to Write Multi-Org Enabled Views

To write multi-org-enabled views, you need to find the view's original definition from the database, and change the views referenced inside the original definition to grammar like this:

```
SELECT . . .
  AP_CHECKS_PKG.GET_POSTING_STATUS(AC.CHECK_ID) POSTING_FLAG
FROM #{AP_BANK_ACCOUNTS} ABA,…
WHERE . . .
```

That is, change "`FROM AP_BANK_ACCOUNTS ABA`" to "`FROM #{AP_BANK_ACCOUNTS} ABA`".

If the organization id list is "204, -1", the meaning of the grammar is as follows:

Grammar	Translated into
`#{abc}`	`(select * from abc_ALL where org_id in (204,-1))`
`#{abc|real_table_name_ALL_B}`	`(select * from real_table_name_ALL_B where org_id in (204,-1))`
`#V{abc}`	`'(' + content from abc.sql + ')'`
`#L{the_column}`	`the_column in (204,-1)`

The `#V{abc}` is for recursive translated view definitions.

Using the Salesforce.com Adapter

Salesforce.com provides a flat, single-level hierarchy of resources. The resources are called SObjects in Salesforce.com terminology. SObjects are represented as relational tables in TDV.

Introspecting resources is time-intensive, so as a general rule select only the resources necessary for your project. If you find you need additional resources later, you can always add them using the Add/Remove Resources in Studio.

This topic describes how to use the Salesforce.com data source adapter with TDV.

Salesforce.com Limitations

Query Limitation

Salesforce.com has limitations on queries. Performance issues will occur if you do NOT restrict the number of results.According to the Salesforce.com documentation, the expected limit on query results is 2,000. However, that limit is subject to change with each version of Salesforce.com, you are responsible to determine what the current limit is for your version of Salesforce.com and adjust your query limits as necessary.

Queries to Salesforce need to include a WHERE clause that restricts the number or records to that limit. For example, 2,000. Otherwise, the queries can time out or performance is adversely affected.

Application Views for Salesforce.com

The organization of the pre-built application views and folders in the Salesforce.com Adapter is patterned after the user interface of an uncustomized Salesforce.com account. Each folder within the DataServicesForSalesforce.com folder corresponds to a tab in Salesforce.com. Each view within the folder corresponds to one of the default views in Salesforce.com.

Filter Data from Application Views

There are two ways to filter data from Salesforce.com using application views:

- using a filter provided by Salesforce.com itself
- by filtering within TDV.

For more information about filters, refer to TDV SQL Support for Salesforce.com, page 50.

Salesforce.com Basic Tab

For Connection Information on the Basic tab, you might need to use a secure password to access the Salesforce.com data source. See Salesforce.com Identity Confirmation Security Feature, page 48 for how to do this.

Salesforce.com Advanced Tab

Click the Advanced tab to display a panel of advanced properties for the data source. These properties are specific to TDV and how it interacts with Salesforce.com:

Advanced Tab Field	Description
	Timeout for the HTTPS connection to Salesforce.com. The default is 60 seconds. This is not a session timeout. Salesforce.com session timeout (default 120 minutes) is configurable at Setup | Security Controls on the Salesforce.com website.
	The maximum number of concurrent calls for the current user. The default is 3.

Advanced Tab Field	Description
	The maximum number of concurrent calls for this data source. The default is 5. This setting prevents TDV from emitting too many concurrent requests to Salesforce.com. For example, if set at 5 but there are 20 views scheduled to trigger concurrently, TDV will queue 15 views. Typically, this limit should not be increased, but might be decreased in case there are several Salesforce.com data sources configured.
	The maximum number of rows returned from a single request to Salesforce.com. The default is 500 rows. The maximum query batch size is 2,000 records; however the maximum is automatically reduced if you select large text fields or many fields. This behavior is implemented by Salesforce.com on their servers to maintain performance. Even if a query batch size of 200 is selected, Salesforce.com reserves the right to return more or fewer records. For example, the batch size will be no more than 200 if a query selects two or more custom fields of type long text. This is to prevent large SOAP messages from being returned. The Salesforce.com driver for TDV makes as many requests as necessary to Salesforce.com to fully evaluate a query, regardless of the number of rows returned per request by Salesforce.com. For example, if the batch size is set to 2,000 rows (and assuming Salesforce.com honors this number of rows) but there are 4,000 rows in the object and a query spans all of them, TDV issues two requests to Salesforce.com.
	Since Salesforce.com is a collection of Web services accessed over the Internet using HTTPS, connections are inherently unreliable. They may timeout or fail altogether. To improve reliability of queries built upon this infrastructure, the Maximum retries can be set to automatically retry any Salesforce.com request upon connectivity failure. The default is 0, meaning the feature is disabled and no retries are attempted.
	If the maximum retries feature is enabled, this property specifies the number of seconds to wait between retry attempts. It defaults to 0.
	If this checkbox is selected, all requests to Salesforce.com for this data source are passed through a Web proxy specified by the properties Proxy host, Proxy port, and optionally Proxy username and Proxy password.
	Host name or IP address of Web proxy. The proxy is used only if the Proxy enabled property is selected.
	Port of Web proxy. The proxy is used only if the Proxy enabled property is selected.

Advanced Tab Field	Description
	Specifies the username and password to be used to authenticate to the selected Web proxy.
	Salesforce.com limits the number of records that can be impacted with a single update request to 200. If this option is disabled (the default), Salesforce.com throws an exception if this limit is exceeded. If this option is enabled, updates impacting more than 200 rows are batched to Salesforce.com to workaround the limit. The batches are not transactional, so if the first batch succeeds but the second fails, the first batch still causes an update of Salesforce.com records.
	Salesforce.com limits the number of records that can be impacted with a single delete request to 200. If this option is disabled (the default), Salesforce.com throws an exception if this limit is exceeded. If this option is enabled, deletes impacting more than 200 rows are batched to Salesforce.com to workaround the limit. The batches are not transactional, so if the first batch succeeds but the second fails, the first batch still causes a delete of Salesforce.com records.
	This property specifies the location of the Salesforce.com environment. It contains two preset values and allows custom values to be provided. By default it is set to the first preset value in the list, the URL of the production Salesforce.com environment. It can also be set to the Salesforce.com Sandbox (the second preset URL), or to the URL of a customer-provided, system on the premises. Typically the SFDC URL is similar to: https://cs5.salesforce.com/

5. Expand the node representing the Salesforce.com data source, and select the boxes corresponding to the resources.

Salesforce.com Identity Confirmation Security Feature

Salesforce.com has provided an additional identity confirmation security feature that might be required to access the Salesforce.com data source. It is possible that you need to implement this additional feature if you receive this error:

```
SForceException: Error [sforce-2900000]:
LOGIN_MUST_USE_SECURITY_TOKEN
```

To access Salesforce.com through a desktop application or other API-based application, you must replace your current password with a combination of your password and a security token.

To implement the Salesforce.com security feature

1. Log in to Salesforce.com through the browser to request your security token.
2. At the top of any Salesforce page, click the down-arrow next to your name. From the menu under your name, select Setup or My Settings—whichever one appears.
3. Go to Setup > My Personal Information > Reset Security Token.

   or

   Go to My Settings > Personal > Reset My Security Token.
4. Click the Reset Security Token button to trigger an email that contains your security token.
5. Select and copy the token from the email.
6. In the application, replace your password with combination of the password and the security token. For example, if your password is MyPassword and your security token is XXXXXX, you would enter MyPasswordXXXXXX in the password field.

See http://trust.salesforce.com/trust/security/identity_feature.html for more details.

Security for the Salesforce.com Adapter

Protecting Salesforce.com Resources

TDV supports the complete set of data security measures offered by Salesforce.com. Security can be configured at a variety of levels: per user, object, field within object, or operation (such as INSERT). All security features are configured using the Salesforce.com Web site's administration screens, and become effective immediately to TDV users.

You can use Salesforce.com's identity confirmation security feature with your password, as described in Salesforce.com Identity Confirmation Security Feature, page 48.

With pass-through security, you can avoid exposing a Salesforce.com Administrator's access rights to TDV users. Users simply log in to TDV with their Salesforce.com user credentials, which are forwarded to Salesforce.com. This ensures that the proper Salesforce.com data security is applied to every user, and that credentials are never stored within TDV.

For information about setting pass-through login, see the *TDV User Guide*.

TDV SQL Support for Salesforce.com

Salesforce.com exposes functionality through a set of SOAP RPC services operating on a namespace of data objects called SObjects. Leveraging these services and a query language called SOQL (Sforce Object Query Language), the Salesforce.com Adapter provides SQL access to Salesforce.com as a relational data source.

The following sections describe how Salesforce.com resources operate within TDV. This topic describes how TDV interprets and supports Salesforce.com SQL statements and objects.

Introspection

Resource Hierarchy

The Salesforce.com Adapter provides Salesforce.com SObjects as resources within TDV. SObjects are introspected by selecting them by name from a flat name space. If you're not sure which SObject contains the data you need, examine the tabs on the Salesforce.com Web site or consult its documentation.

Metadata Mapping

Each field of the Salesforce.com SObject becomes a column in TDV with the same name. Each data type in Salesforce.com is mapped to a TDV SQL-based type. The following table lists the Salesforce.com data types, whether they are supported in TDV, and their corresponding TDV types.

Salesforce.com Type Name	Supported?	TDV Data Type
STRING, TEXTAREA, PHONE, URL, EMAIL, COMBOBOX, PICKLIST, MULTIPICKLIST	Yes	VARCHAR
BOOLEAN	Yes	BIT
INT	Yes	BIGINT
DOUBLE, CURRENCY, PERCENT	Yes	DECIMAL, NUMERIC
DATE	Yes	DATE
DATETIME	Yes	TIMESTAMP
BASE64	Yes	VARCHAR
ID, REFERENCE	Yes	VARCHAR(18)

Capabilities

Capabilities are a system of classifying the unique features and limitations of data sources. For example, an Oracle data source can execute subqueries, but Salesforce.com cannot. Capabilities are consulted when a query is processed so that data sources receive only the query processing work they support. Where a capability is lacking in a data source but required to run a query, TDV performs the work itself.

The following table lists commonly used capabilities and how they apply to Salesforce.com. The Supported column indicates whether or not the SQL capability is supported in queries against Salesforce.com. The Pushed column indicates whether the capability is supported directly on Salesforce.com, allowing the query processing work to be off-loaded to it. For efficient queries, minimize use of non-pushed capabilities.

Capability	Supported?	Pushed?	Notes
CASE	Yes	No	
DELETE	Yes	Yes	See Data Modification in Salesforce.com, page 59
DISTINCT	No	No	
Filter	Yes	Yes	Filters comparing two columns of the same SObject are not allowed by Salesforce.com. If a filter is applied to a column that has the "filterable" attribute set to false, the query is executed in TDV instead of being pushed to the Salesforce data source.
Filter – BETWEEN	Yes	Yes	
Filter – IN	Yes	Yes	See Multipicklists with SQL, page 56
Filter – LIKE	Yes	Yes	
Functions – Aggregate	Yes	No	
Functions – CAST	Yes	Yes	
Function – ConvertCurrency	Yes	Yes	Salesforce.com-specific function. Currency management must be enabled in Salesforce.com for this to function properly. Salesforce.com constraints on the use of ConvertCurrency apply. See Salesforce.com documentation for more information.
Functions – Others	Yes	No	
Function – ToLabel	Yes	Yes	Salesforce.com-specific function.
GROUP BY	Yes	No	

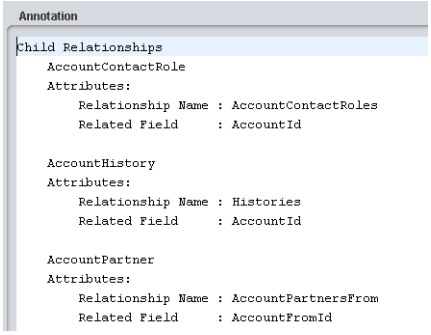
Capability	Supported?	Pushed?	Notes
INSERT	Yes	Yes	See Data Modification in Salesforce.com, page 59
JOIN	Yes	Yes	See Joins, page 53 for details about join types that are pushed.
ORDER BY	Yes	No	
Subquery	Yes	No	
Transactions	No	No	
UNION	Yes	No	
UPDATE	Yes	Yes	See Data Modification in Salesforce.com, page 59

Joins

When it is possible to optimize the performance of queries with joins, TDV uses any join information it finds to convert standard SQL to SOQL and pushes this to the source Salesforce.com database at run time (that is, when data is actually retrieved). Specifically, queries with inner and outer joins that have parent-to-child or child-to-parent relationships are pushed to Salesforce.com at run time. If these relationships do not exist, the joins are processed in TDV at run time.

Note: You can disable pushing joins to Salesforce.com. See Disabling Pushing Join Execution, page 55. If you import a CAR file with Salesforce.com resources from a TDV release prior to 6.2 SP2, you must reintrospect the resources to take advantage of the join pushing capabilities.

Join relationships are evaluated when a Salesforce.com data source is created and introspected. If any parent-to-child or child-to-parent relationships are discovered in any tables during introspection, the child relationships are collected for each table and inserted in the Annotation field :



This information is for user information only and cannot be edited. If the Annotation field is empty, no child relationships exist for this table.

Parent relationships are displayed as foreign key relationships in TDV in the table's Foreign Keys tab. The foreign and primary key relationships are used at run time.

Joins Pushed to Salesforce.com

TDV can push right outer joins, left outer joins, and inner joins to Salesforce.com. Examples of how TDV translates these joins from SQL into SOQL are provided below.

Right Outer Join

Find all opportunity records and their related accounts.

SQL:

```
SELECT
o.Id, o.Name,
A.Id, A.Name
FROM /shared/QA_SForce/Sources/SForce/Account a right outer join
/shared/QA_SForce/Sources/SForce/Opportunity o
on a.id = o.AccountId
```

SOQL:

```
Select Id, Name, Account.Id, Account.Name
From Opportunity
```

Left Outer Join

Find all the accounts and their related opportunity records.

SQL:

```
SELECT
A.Id, A.Name,
o.AccountId, o.Name
FROM /shared/QA_SForce/Sources/SForce/Account a left outer join
/shared/QA_SForce/Sources/SForce/Opportunity o
on a.id = o.AccountId
```

SOQL:

```
Select Id, Name, (Select  AccountId, Name From Opportunities)
From Account
```

Inner Join

Find all accounts where there exists an opportunity record.

SQL:

```
SELECT
A.Id, A.Name
FROM /shared/QA_SForce/Sources/SForce/Account a inner join
/shared/QA_SForce/Sources/SForce/Opportunity o
on a.id = o.AccountId
```

SOQL:

```
Select Id, Name
From Account
Where Id In
(Select AccountId from Opportunity)
```

Disabling Pushing Join Execution

TDV is configured to automatically push any joins it can to Salesforce.com. You can disable pushing join execution to Salesforce.com this way:

- Use the DISABLE_PUSH option in your query, as in this example:

```
SELECT {option DISABLE_PUSH}
o.Id, o.Name,
A.Id, A.Name
FROM /shared/QA_SForce/Sources/SForce/Account a right outer join
/shared/QA_SForce/Sources/SForce/Opportunity o
on a.id = o.AccountId
```

Semijoins

A semijoin is the best way to reduce the number of Salesforce.com rows retrieved and processed by TDV, thus improving query performance. To force a semijoin to occur in a query, add it immediately before the table to be joined. For example:

```
SELECT * FROM A INNER JOIN { option semijoin } B ON A.K = B.K
```

Values of A.K are collected and passed in a query to B as the filter.

```
SELECT * FROM A
SELECT * FROM B WHERE K IN ({values of A.K from previous query})
```

If there are many rows in A, this can result in lengthy queries against B. TDV automatically partitions the queries against B if they become too large. Partitioning means the query is broken up into smaller queries that are executed separately and reassembled to produce a unified result set.

Put the larger table on the right side of the join whenever possible. When running a new query for the first time, display the Execution Plan in Studio (click Show Execution Plan in the resource editor) and then click Execute and Show Statistics. Examine each node's row count and query after processing has begun to make sure filters are pushed down to Salesforce.com. This is a good way to see the mechanics of a semijoin in action. If the interaction between TDV and Salesforce.com is still unclear and performance is poor, enable debug logging as described in the *TDV Installation and Upgrade Guide*. This can help illustrate how a SQL statement on TDV becomes a series of requests to Salesforce.com.

Multipicklists with SQL

Multipicklists are a data type within Salesforce.com which allow the user to select multiple values from a list of valid values. When selected in SQL queries within TDV, Multipicklists are displayed as a series of values delimited by semicolons. For example, if the user has selected Bob and Mary from a Multipicklist, the value visible to TDV is `Bob;Mary`.

To treat the selected values as separate for the purposes of filtering, Salesforce.com provides special operators to their query language. This section describes how to structure your SQL query in TDV to leverage these operators.

Equal

When the Multipicklist field uses the equality operator, it is converted to the includes operator when issued to Salesforce.com. For example, the following SQL statement returns rows where the field Multipicklist__c includes Value1 in its selection:

```
SELECT * FROM TestObject1__c
WHERE MultiPickList__c = ‘Value1’
```

This query results in the following filter on the query issued to Salesforce.com:

```
WHERE MultiPickList__c includes (‘Value1’)
```

Not Equal

When the Multipicklist field uses the inequality operator, it is converted to the NOT and includes operators when issued to Salesforce.com. For example, the following SQL statement returns rows where the field Multipicklist__c includes Value1 in its selection:

```
SELECT * FROM TestObject1__c
WHERE MultiPickList__c != ‘Value1’
```

This query results in the following filter on the query issued to Salesforce.com:

```
WHERE NOT (MultiPickList__c includes (‘Value1’))"
```

Includes

A Multipicklist field includes a value if the field’s selection includes the value among its selected values. The IN keyword is used to instruct TDV that the INCLUDES syntax should be pushed to Salesforce.com. For example, the following SQL statement returns rows where the field Multipicklist__c includes Value1 and Value2 in its selection:

```
SELECT * FROM TestObject1__c
  WHERE MultiPickList__c IN ('Value1', 'Value2')
```

This query results in the following filter on the query issued to Salesforce.com:

```
WHERE MultiPickList__c includes ('Value1','Value2')
```

Excludes

A Multipicklist field excludes a value if the field’s selection does not include the value among its selected values. The NOT IN keyword is used to instruct TDV that the EXCLUDES syntax should be pushed to Salesforce.com. For example, the following SQL statement returns rows where the field Multipicklist__c does not include Value1 and Value2 in its selection:

```
SELECT {OPTION DISABLE_PUSH}
*
FROM TestObject1__c
  WHERE MultiPickList__c NOT IN ('Value1', 'Value2')
```

Note: The DISABLE_PUSH option is required to ensure that Salesforce.com does not return NULLs.

This query results in the following filter on the query issued to Salesforce.com:

```
WHERE MultiPickList__c excludes ('Value1','Value2')
```

Lead Conversion

Salesforce.com provides a ConvertLead API call which converts a Lead into an Account, Contact, or Opportunity. The stored procedure named convertLead within the Salesforce.com Data Source can be leveraged to access this functionality.

To convert a Lead, the Salesforce.com data source must be logged into Salesforce.com with Convert Leads permission and Edit permission on Leads, as well as sufficient permissions to create or update, as applicable, any associated Account, Contact, and Opportunity.

The details involved in converting a lead are outside the scope of this document. For more information, see the Salesforce API documentation in the Developer section of the Salesforce.com Web site.

Updated and Deleted Objects

Salesforce.com provides stored procedures named GetUpdated and GetDeleted, which are API calls that return IDs of specified objects updated or deleted between dates in a specified range.

To be accessed through GetUpdated and GetUpdated, an object must be configured so that it can be replicated. Results are returned for no more than 30 days previous to the day the call is executed. They need to return the IDs of Contacts that have been updated or deleted on the current date.

Upsert

The Upsert API call creates new records and updates existing records. Use Upsert instead of Create to avoid creating unwanted duplicate records.

Upsert uses a custom field to determine the presence of existing records. All records must be of the same object type. Maximum array size is 200.

Inputs

externalIdFieldName: Determines whether it should create a new record or update an existing one.

objectType: The record type.

dataSql: SQL Used to generate the new records. Each row in the resulting result set corresponds to a record in SalesForce. The dataCursor input can be used instead.

dataCursor: Input cursor used to generate the new records. Each row in the resulting result set corresponds to a record in SalesForce. The dataSql input can be used instead.

fields: Comma-separated list of fields in the record. If this is NULL, fields are extracted from the dataSql or dataCursor if possible. Otherwise, an error occur.

Outputs

Upsert outputs a cursor containing the Upsert results:

id: Salesforce ID of the record.

isCreated: Boolean indicating if the record was newly created or not.

Merge

Merge merges records of the same object type into one of the records, deletes the others, and adjusts the parenting of related records.

Inputs

objectType: The master record type. The only supported types are Lead, Contact, and Account.

masterRecordId: The ID of the master record that others are merged into.

recordToMergeIds: Comma-separated list of the records to merge into the master record.

Outputs

A cursor containing the merge results.

Packaged Query

TDV supports packaged queries against Salesforce.com using SOQL.

This support allows for the building of views on top of the packaged queries to flatten the nested results. If relevant metadata is available, it is pushed down using joins.

Data Modification in Salesforce.com

INSERT, UPDATE, and DELETE operations can be performed directly against any Salesforce.com data source. In TDV, this is done through a SQL Script. For details on SQL scripts, see the *TDV Reference Manual*.

Data modification in Salesforce.com is governed by a set of business rules often defined on a per SObject basis, which are beyond the scope of this document to address in depth. This section contains a summary of some rules, paraphrased from the Salesforce.com API documentation.

If a data modification request to Salesforce.com is not processed as expected, it may be due to a business rule on the Salesforce.com server. For more information about the specifics of updating Salesforce.com SObjects, consult the Salesforce API documentation, available in the Developer section of the Salesforce.com Web site.

- INSERT INTO, page 60
- UPDATE, page 61
- DELETE, page 62

INSERT INTO

The INSERT INTO statement creates a new SObject instance of a particular type. The type is determined by the resource provided as the table to the command. Following the resource name, a list of field names and values provide the data to initialize the new object.

For example:

```
INSERT INTO
/shared/DataServicesForSalesforce.com/Sources/Salesforce.com/Conta
ct
(FirstName, LastName, MailingStreet, MailingCity, MailingState,
MailingPostalCode,
MailingCountry, Phone)
VALUES ('Marc', 'Benioff',
'1 Market St Suite 300', 'San Francisco', 'CA', '94105', 'US',
'415-555-1212');
```

Below are a few of the areas to consider while developing an INSERT INTO statement:

- Account security. The Salesforce.com account used by TDV must have sufficient access rights to create objects of the specified type.
- Object and field security. Some objects and certain fields within those objects require special handling or permissions. For example, you might also need permissions to access this object's parent object. Some objects cannot be created through the API.
- Generated fields. Salesforce.com generates unique values for some fields automatically. You cannot explicitly specify an ID, CreatedDate, CreatedById, LastModifiedDate, LastModifiedById, and SystemModstamp.
- Required fields. For required fields that do not have a preconfigured default value, you must supply a value.
- Default values. For some objects, some fields have a default value. If you do not specify a value for such fields, Salesforce.com populates these fields with the default value.

- Referential integrity. If you are creating an object that is the child of a parent object, you must supply the foreign key information that links the child to the parent.
- Number of records inserted at a time. When doing an insert into Salesforce.com, TDV always uses batch inserts of no more than 200 records at a time.

UPDATE

The UPDATE statement changes one or more SObject instances, based on a filter and list of fields and values. The SObject type is determined by the resource provided as the table to the command. Following the resource name, a list of field-value pairs detail the changes to be performed on the selected objects. Finally a filter specifies the conditions to be satisfied for an update to take place on a given row.

For example:

```
UPDATE
/shared/DataServicesForSalesforce.com/Sources/Salesforce.com/Conta
ct
SET "Phone" = '415-555-9999'
WHERE "FirstName" = 'Marc' AND "LastName" = 'Benioff';
```

Below are a few of the areas to consider while developing an `UPDATE` statement:

- Account security. The Salesforce.com account used by TDV must have sufficient access rights to update objects of the specified type.
- Object and field security. Some objects and certain fields within those objects require special handling or permissions. For example, you might also need permissions to access this object's parent object. Some objects cannot be updated through the API.
- Generated fields. Salesforce.com generates unique values for some fields automatically. You cannot explicitly update an ID, CreatedDate, CreatedById, LastModifiedDate, LastModifiedById, and SystemModstamp.
- Required fields. When updating required fields, you must supply a value—you cannot set the value to null.
- Referential integrity. Fields whose names contain "Id" are either that object's primary key or a foreign key. Salesforce.com does not allow the update of primary keys, but foreign keys can be updated.
- Object change limit. Salesforce.com allows up to 200 objects to be changed in a single request. If an update request exceeds 200 objects, the entire operation fails (unless the Advanced data source property Batch Updates is enabled).

DELETE

The DELETE statement removes one or more SObject instances based on a filter. The SObject type is determined by the resource provided as the table to the command. Following the resource name, a filter specifies the conditions to be satisfied for a delete to take place on a given row. For example:

```
DELETE FROM
/shared/DataServicesForSalesforce.com/Sources/Salesforce.com/Conta
ct
WHERE "FirstName" = 'John' AND "LastName" = 'Doe';
```

These are a few of the areas to consider while developing a DELETE statement:

Area	Consideration
Account security.	The Salesforce.com account used by TDV must have sufficient access rights to delete objects of the specified type.
Object and field security.	You may need permissions to access this object's parent object. Some objects cannot be deleted through the API.
Referential integrity.	To ensure referential integrity, the DELETE call supports cascading deletes; that is, if you delete a parent object, you delete its children automatically, as long as each child object can be deleted. If the advanced data source property Batch Deletes is enabled, requests exceeding 200 objects can be handled.

Using the SAP for TDV Adapter

This topic describes how to create and introspect an SAP data source.

About SAP Data Sources

For installation requirements, see the *TDV Installation and Upgrade Guide*.

For further details on configuring relational data sources see the *TDV User Guide*.

The SAP data source provides access to four types of resources: ABAP Queries, Functions, InfoSets, and Tables:

- ABAP Queries are organized by query area and functional area.
- You can browse functions using two hierarchies: BAPI and RFC. The BAPI hierarchy organizes functions are organized as in the SAP Business Object Repository. The RFC hierarchy organizes functions by SAP Development Class and Function Group.
- InfoSets are organized by query area.
- Tables are organized by the first two characters of their name.

Note: If the SAP system is Unicode, RFC and BAPI resources support Unicode. However, such support for table resources requires SAP Note 758278.

Application Views for SAP

The organization of the views and folders mirrors that of SAP's Business Object Repository (BOR). Field names are aliased to provide human-readable names. In some cases several objects in BOR are synthesized in a single query to produce a more detailed, unified view of the data.

A common pattern employed in application views is that of List and Details. Views with names ending in "List" return a minimal set of columns that serve to uniquely identify an object. The identifiers are passed as arguments into a corresponding view, its name ending in "Details," to produce a more extensive set of columns.

Filter Data from Application Views

There are two ways to filter data from SAP using application views: using a filter provided by SAP itself, or by filtering within TDV. Both methods are used by application views depending on the view and the capabilities of SAP.

SAP Functions

In this document, the word "function" means any RFC-enabled function in SAP. The resources located in the BAPI and RFC folders are all functions for the purposes of query processing.

SAP functions are represented as table resources in TDV to aid interoperability with SQL, but the best way to think of an SAP function is as a stored procedure, with inputs and outputs. Invoking it is a discrete event. SAP functions can have scalar, structure, and table-type parameters. In TDV these are mapped to a single virtual table representing the SAP function. To call SAP functions within SQL, TDV maps the SQL WHERE clause to SAP function inputs, and SAP function outputs are mapped to relational columns.

SAP Tables

SAP tables resemble ordinary relational database tables, with one or more primary keys. SAP foreign keys are not visible in TDV. TDV interprets SQL table queries and translates them into SAP API calls. For joins. TDV pulls necessary data from SAP and processes the query outside of SAP, slowing TDV and SAP performance.

ABAP Queries

ABAP queries are views defined within SAP. An ABAP query in TDV consists of columns with no primary key defined. Columns with names starting with an underscore character are input parameters to the query, and can be used to filter data.

- InfoSets in TDV perform identically to ABAP queries, except that they are not divided by functional area, and during introspection an InfoSet query (ABAP query) is generated in SAP by TDV.

Use in a load balanced environment

If you are using TDV and SAP in a load balanced environment, then you must specify these fields when defining the SAP data source:

- On the Basic tab, specify Client, User, Password, and Language. Application Server, SAP Router String, and System Number are not applicable.
- On the Advanced tab, specify Message Server, System ID, and Logon Group.

When setting up your SAP data source you can define pass-through login information. The operations you can and cannot perform in pass-through mode depend on whether or not Save Password is checked.

Save password ?	Operations you can perform	Operations you cannot perform
Yes	Introspection. You do not have to resupply the password.	N/A
No	• Query/update/insert/delete operations. You need to resupply the original login credentials for the current session. • Re-introspection, Add/Remove data source resources. You will be prompted to resupply the password that was used when the data source was originally introspected.	Schedule reintrospection.

SAP Basic tab

Field	Description
Application Server	Name of the machine hosting SAP or the host machine's IP address. For load-balanced configurations of SAP, leave this property empty.
SAP Router String	Routing entry. If an SAP Router is used to connect to the Application Server, enter its routing string here. Sample: /H/saprouter/H/194.117.106.130/S/3297/H/
System Number	Two-digit system or gateway service number of the SAP system. For load-balanced configurations of SAP, leave this property empty.
Client	Three-digit client number of the SAP system.
User and Password	Valid user name and password to SAP.
Save Password	This option works in combination with the Pass-through Login option. By default, this option is disabled and not editable. It becomes editable when you select the Pass-through Login option. For more details, see the *TDV User Guide*.
Pass-through Login	Works in combination with the Save Password option. By default, this mode is disabled ("non-pass-through mode") and the password is saved. Refer to the description for the Save Password option (above). If you select the Enabled option, the Save Password option becomes editable.
Language	The SAP logon language. Default is EN (English).

SAP Advanced tab

Field	Description
Maximum connections in pool	SAP JCo parameter specifying the maximum number of simultaneous connections in an SAP connection pool.

Field	Description
Maximum idle connections in pool	SAP JCo parameter specifying the maximum number of simultaneous idle connections in an SAP connection pool kept open by the destination. A value of 0 has the effect that there is no connection pooling. That is, connections will be closed after each request.
Table row count	Maximum number of records to retrieve from SAP for any query against a Table resource. The default is 250,000. If queries are taking too long to run, setting this value to 1,000 can help troubleshoot the issue and prevent "runaway queries." Setting this value to 0 means there is no restriction on the number of rows returned. TDV must be restarted for a change to table row count to take effect.
Function thread pool size	Maximum number of threads available to invoke SAP functions in parallel for a single query. Setting this property to a value greater than 0 enables TDV to make function calls in parallel to SAP. This can improve performance when many function calls are required to process a SQL query. For example, if a table containing 10 customer IDs is joined with an SAP function to retrieve customer details by ID, the function will be invoked 10 times. Setting this property to 5 would result in 5 of the function calls being executed simultaneously. The default is 0, which means this feature is disabled and function calls are processed serially.
ABAP Query DB access limit	Maximum number of database accesses SAP allows during execution of queries against ABAP Queries and InfoSets. This does not correlate to number of rows returned, because database access patterns vary significantly between ABAP Queries. In general, reducing the database access limit results in fewer rows returned and provides faster response times. Valid values range from 1 to 99,999,999. The default is 1,000.
Table read function	An SAP function to be used for accessing table resources. Default is RFC_READ_TABLE.
Table row length	Length of data field returned by table read function, in bytes. Default is 512.
Table decimal fix	If checked, SAP Note 758278 or equivalent has been applied to SAP such that the table read function encodes decimal values properly. Default is unchecked.

Field	Description
Table maximum options	Maximum number of filter options that can be sent to SAP when querying a table resource. Default is 500. Filter options are created from the WHERE clause for SQL queries, so this setting is important when tuning queries containing semijoins. The number of options allowed by SAP depends on its version and configuration. Too many options typically results in a core dump error from SAP.
Load-balance configuration options	
Message Server	Host name of the SAP Message Server.
Message Server Port	Port of the SAP Message Server.
System ID	Logical name of the SAP system, also known as R3Name.
Logon Group	Represents a group of SAP Application Servers.

Configuring TDV Logging Level when Using SAP for TDV

The logging level that you set for TDV can have an impact on your disk space. Because of the way SAP and TDV interact, if you set the TDV logging level to the highest level (debug), log files are written to the SAP Adapter's root directory. The files are created for all successful and unsuccessful actions. Capturing this much information can cause your system to run out of space.

To set the logging level

1. Locate the TDV log4j.properties file.
2. Make sure that the following line is commented out (that is, have a hash sign at the beginning):

```
#log4j.logger.com.compositesw.cdms.ds.sap=DEBUG
```

3. Locate and open the product.properties file.
4. Make sure that sap.debug is set to false.

5. Locate the following properties in the product.properties file. Typical installations use the default values of these properties.

Property	Description
sap.debug.freespaceLimitInMB =100	If the usable file system space goes beyond the configured limit, no new debug log files are written. This should prevent file system capacity exhaustion.
sap.debug.cleanUpAtStart=false	When set to true, the first attempt to write one of the debug files, triggers an attempt to clear earlier log files from that directory.

SQL Support for SAP

This section describes TDV SQL support for SAP—SAP resource types (functions, tables, and ABAP queries) and their behavior within TDV SQL queries.

- TDV SQL Support for SAP Functions, page 69
- TDV SQL Support for SAP Tables, page 77
- TDV SQL Support for SAP ABAP Queries, page 81
- TDV SQL Support for SAP Dates, page 84

TDV SQL Support for SAP Functions

The following section describes how SAP functions work within TDV. It is divided into five parts:

- Introspection and SAP Functions, page 69
- Capabilities and SAP Functions, page 71
- Parameter Mapping and SAP Functions, page 73
- Result Mapping and SAP Functions, page 75
- Joins and SAP Functions, page 77

Introspection and SAP Functions

Introspection examines SAP for its list of functions, allows the user to select functions, and translates the function metadata to relational tables that TDV understands. The two sections below explain:

Resource Hierarchy

The introspector contains two folders for SAP functions: BAPIs and RFCs. Each folder shows the same set of functions at the leaf nodes, but the hierarchy differs. BAPI hierarchy consists of Applications, Object Types, and Object Methods, which are the functions themselves. RFC hierarchy contains Development Classes, Function Groups, and the functions themselves.

To use an SAP function in TDV it must be RFC-enabled, and the user specified in the SAP data source configuration must be authorized to invoke RFCs. SAP transaction SE37 can be used to determine if a function is RFC-enabled (a "Remote-Enabled Module").

Metadata Mapping

To call an SAP function, the following information is needed:

- Function name
- Import parameters (collection of named input fields)
- Export parameters (collection of named output fields)
- Table parameters (collection of named tabular structures)

In TDV, all import, export, and table parameters become columns in a single, named virtual table that represents the SAP function. The following naming convention is used to map fields of an SAP function to columns of a TDV table:

  - — Class—Import or Export. Omitted if the field is a table parameter.
  - — Structure—Name of the structure containing the value. Omitted if the value is scalar. Table structure name if the field is a table parameter.
  - — Field—Required name of the field.

Each data type in SAP is mapped to a TDV SQL-based type. TDV is limited to types supported in the underlying SAP JCo library.

The following table lists the SAP data types, whether they are supported in TDV, and their corresponding TDV type.

SAP Data Type	ABAP Type	Supported	TDV Data Type
CHAR, UNIT, CUKY, CLNT, LANG, LCHR, VARC	C	Yes	VARCHAR
DATS	D	Yes	DATE
FLTP	F	Yes	DOUBLE
STRING	g	Yes	VARCHAR
INT4, INT2, INT1	I	Yes	BIGINT
NUMC	N	Yes	VARCHAR
CURR, QUAN, DEC, PREC	P	Yes	DECIMAL, NUMERIC
TIMS	T	Yes	TIME
LRAW, RAW	X	Yes	VARCHAR
XSTRING	y	Not supported. Cannot be used in RFC-callable functions.	

Documentation of SAP function and its columns is retrieved from SAP and stored in the Annotation field in TDV.

Capabilities and SAP Functions

Data-source-specific capabilities determine how a SQL statement is to be divided between SAP, TDV, and other data sources that might be referenced in the query. Where a capability is lacking in a data source but required to run a query, TDV typically compensates by performing the work itself.

The following table lists commonly used capabilities and whether they are supported in queries against SAP functions. Pushed indicates whether the capability is supported directly on SAP. For efficient queries, it is best to minimize use of capabilities that cannot be pushed.

Capability	Supported?	Pushed?	Notes
CASE	Yes	No	
DELETE	No	No	Every SAP function handles deleting data differently. Consult SAP documentation before constructing SELECT statements with DELETE.
DISTINCT	Yes	No	
Filter	Yes, with special usage	Yes	All filters are mapped to parameters. (See Parameter Mapping and SAP Functions, page 73.) SAP does not filter output parameters, but by default they are pushed anyway. Workaround: wrap queries to SAP functions.
Filter–BETWEEN	Yes, with special usage	No	See Notes on Filter.
Filter–IN	Yes, with special behavior	Yes	See Parameter Mapping and SAP Functions, page 73.
Filter–LIKE	Yes, with special usage	No	See Notes on Filter.
Functions–aggregate	Yes	No	
Functions–CAST	Yes	Yes	
Functions–others	Yes	No	
GROUP BY	Yes	No	
INSERT	No	No	Every SAP function handles inserting data differently. Consult SAP documentation before constructing SELECT statements with INSERT.
Join	Yes	No	

Capability	Supported?	Pushed?	Notes
ORDER BY	Yes	No	
Subquery	Yes	No	
Transactions	No	No	INSERT/UPDATE/DELETE not supported.
UPDATE	No	No	Every SAP function handles updating data differently. Consult SAP documentation before constructing SELECT statements with UPDATE.

Parameter Mapping and SAP Functions

Parameter mapping is the process of applying SQL filter expressions to the SAP function call. The SQL filter becomes a series of arguments to the SAP function. How this is done depends on the number and types of parameters:

- No Parameters, page 73
- Import Parameters, page 73
- Table Parameters, page 74
- Multiple Parameter Sets, page 74

No Parameters

The following is a valid SQL statement:

```
SELECT * FROM
/Shared/DataServicesForSAP_4_6/Sources/SAP/BAPI_CUSTOMER_GETLIST
```

This invokes the SAP function BAPI_CUSTOMER_GETLIST, and all of the values in its import parameters, export parameters, and table parameters are returned. The SAP system on which to execute the function is specified by the data source named SAP in the resource tree folder:

Shared > DataServicesForSAP_4_6 > Sources

Import Parameters

Import parameters are scalar or structure values that represent function input. Import parameters may or may not be required for the function to succeed, depending on the implementation of the function in SAP.

Include a WHERE clause that assigns a value to each import parameter. For example:

```
SELECT * FROM BAPI_COMPANYCODE_GETDETAIL WHERE "Import COMPCODE" =
'1000'
```

Values must be provided using the "=" operator. Comparison operators such as "<" and ">" have no meaning and are ignored.

The IN operator is supported, but its behavior is special. See Multiple Parameter Sets, page 74 for more information.

Note: Import parameters are often used to capture object identifiers, for example Customer ID. Some SAP functions require that identifiers be padded to their full length with leading zeros. For example, SAP might not recognize the value '1000' for a field of type VARCHAR(10) unless it is padded to '0000001000'.

Table Parameters

To provide a table parameter with a single row of data, follow the instructions described in Import Parameters, page 73. For example, this SQL invokes the function BAPI_CUSTOMER_GETLIST with one row of data in the table CUSTOMERRANGE:

```
SELECT * FROM BAPI_CUSTOMER_GETCONTACTLIST WHERE "CUSTOMERRANGE
SIGN" = 'I' AND "CUSTOMERRANGE OPTION" = 'BT' AND "CUSTOMERRANGE
LOW" = '0000000000' AND "CUSTOMERRANGE HIGH" = '9999999999'
```

To provide multiple rows of data, use an IN clause for each column. An IN clause on a table parameter is interpreted as a set of input rows for a given column.

Multiple Parameter Sets

An SAP function can be invoked multiple times with a single SQL statement. This can be helpful when a function returns detail on a single object but the query must return detail data for multiple customers in a single result set.

To support this, parameters passed in a WHERE clause are decomposed into a set of invocations of the SAP function, where each invocation is a set of input values. The behavior of the WHERE clause with SAP functions is described below.

- OR designates a new invocation of the function.
- NOT is ignored.
- An IN clause on an import parameter is expanded and treated as a series of OR expressions.
- NOT IN is treated like IN.

- An IN clause on a table parameter is preserved as a set of input rows for a given column. This enables table parameters to be populated with many rows of data, using an IN clause for each column.

The four examples below illustrate different combinations of the rules and their results. The SQL statement is listed first, and then a breakdown of the function invocations that would result.

1. Single invocation, imports (2 fields)

```
SELECT * FROM FOO WHERE "Imports A" = 1 AND "Imports B" = 2
```

Results in an invocation to FOO with Imports A=1, B=2.

2. Multiple invocations, imports (2 fields)

```
SELECT * FROM FOO WHERE "Imports A" = 1 AND "Imports B" = 2 OR
"Imports B" = 3 AND "Imports A" IN (3, 4)
```

Results in three invocations of FOO with the following values:

```
Imports A=1, Imports B=2
Imports A=3, Imports B=3
Imports A=4, Imports B=3
```

3. Single invocation, imports and a table (1 column, 1 row)

```
SELECT * FROM FOO WHERE "Imports A" = 1 AND "FOO_TABLE B" = 2
```

Results in one invocation of FOO with the following values:

```
Imports A=1, FOO_TABLE (row 1): B=2
```

4. Multiple invocations, imports, and a table (2 columns, 2 rows)

```
SELECT * FROM FOO WHERE "Imports A" = 1 AND "FOO_TABLE B" IN (3, 4)
AND "FOO_TABLE C" IN (5, 6) OR "Imports A" = 2 AND "FOO_TABLE B" IN
(7, 8) AND "FOO_TABLE C" IN (9, 10)
```

Results in two invocations of FOO, with the following values:

```
Imports A=1, FOO_TABLE (row 1): B=3, C=5 (row 2): B=4, C=6
Imports A=2, FOO_TABLE (row 1): B=7, C=9 (row 2): B=8, C=10
```

Result Mapping and SAP Functions

Result mapping is the process of translating data returned by an SAP function call into a standard database result set with rows and columns. Because data structures within SAP functions are more complex than rows and columns, this is not always a one-to-one mapping.

- Import Parameters, page 76
- Export Parameters, page 76
- Table Parameters, page 76

Import Parameters

Import parameters provided in filters are returned unmodified in each row of the result set.

Export Parameters

Export parameters are returned unmodified in each row of the result set.

Table Parameters

An SAP function can contain many table parameters, which are like export parameters with multiple rows. Every table parameter belongs to a named table structure. An SAP function can contain table parameters from multiple, independent table structures, without a common key to relate them. You can simultaneously select table parameters from more than one of these independent table structures.

When a query selects table parameters from more than one table structure, the table structure containing the greatest number of rows is designated as the primary table structure. Values from other table structures are iterated over in lock-step with the primary table structure. When a table structure has no further rows but the primary table structure contains more rows, null values are output.

For example, table structure A contains table parameters A1 and A2 with three rows, and table structure B contains table parameters B1 with one row:

```
SELECT "A A1", "A A2", "B B1"
```

This statement generates the following results:

```
"A A1" "A A2" "B B1"
------ ------ ------
row1-1 row1-2 row1-1
row2-1 row2-2 <NULL>
row3-1 row3-2 <NULL>
```

The behavior is different when table structures and single-row collections (import and export parameters) are combined. These collections always return a single row, so they are repeated over the number of rows returned by the primary table structure. For example, you can have a statement with export parameters E E1 and E E2 and a table structure T containing table parameter T1 with three rows:

```
SELECT "Export E E1", "Export E E2", "T T1"
```

This query returns the following results:

```
"Export E E1" "Export E E2" "T T1"
------------- ------------- ------
val-e1        val-e2        row1
val-e1        val-e2        row2
val-e1        val-e2        row3
```

Joins and SAP Functions

This section describes how to join SAP functions with each other and with other resources. You can use a semijoin to combine the output of an SAP function with another SAP function or any TDV resource. Semijoin uses the results of one query as inputs to another query. As described Multiple Parameter Sets, page 74, every unique input set provided to the SAP function resource causes a function call that adds one or more rows to the result. It may be desirable to control the ordering of the function invocation, and thus the rows in the result, by adding NESTEDLOOP as a query hint; for example, {OPTION SEMIJOIN, NESTEDLOOP}.

If SAP function invocation fails, export parameter values are typically available to diagnose the problem. But from the perspective of the TDV join, the right-side resource still has rows to return, and so NULL values may be injected into the result despite the failure of the function.

With a relational data source, an extra condition would be added to the JOIN expression, so that rows containing an error message would be omitted. Because all filters are pushed to SAP, and SAP only allows filtering function by function, and only via specific table and import parameters, you may need to have TDV take over filtering the results to omit error rows. One way to do this is by writing a SQL script to contain the query against the SAP function, and then a view with which to filter out the NULL rows.

A good idea is to test the SAP function in SAP GUI to understand how it operates, and then test it in isolation in a simple TDV view.

TDV SQL Support for SAP Tables

This section describes how SAP tables work within TDV. It is divided into three parts:

- Introspection and SAP Tables, page 77
- Capabilities and SAP Tables, page 80
- Joins and SAP Tables, page 81

Introspection and SAP Tables

The introspection process examines SAP for its list of tables, allows the user to select tables, and translates the SAP table metadata into relational tables for TDV. The important considerations are:

- Resource Hierarchy, page 78
- Metadata Mapping, page 70

Resource Hierarchy

The SAP Adapter provides logical SAP tables from the SAP data dictionary as resources in TDV. Typically, any table that can be accessed via the SAP function RFC_READ_TABLE can be queried from TDV. Exceptions to this rule are discussed in Metadata Mapping, page 70.

SAP tables are introspected by drilling down into a two-level hierarchy of folders. SAP tables are organized into folders based on the first two characters of their name. For example, the SAP table named VBAK is located in V > VB.

Metadata Mapping

When translating metadata from SAP tables to TDV table resources, each column of the SAP table becomes a column in TDV with the same name. Primary key columns in the SAP table become primary key columns in TDV.

Each data type in SAP is mapped to a TDV SQL-based type that is reported in the FIELDS table returned by RFC_READ_TABLE or YRFC_READ_TABLE. TDV reads the table and uses the field lengths listed there.

Note: TDV recommends using the YRFC_READ_TABLE if it is implemented on your SAP system.

The following table lists the SAP data types, whether they are supported in TDV, and their corresponding TDV type.

SAP Data Type Name	ABAP Type	Supported	TDV Data Type
CHAR, UNIT, CUKY, CLNT, LANG, LCHR, VARC	C	Yes	VARCHAR
DATS	D	Yes	DATE
FLTP	F	No – see below	
STRING, XSTRING	g, y	No – see below	
INT4, INT2, INT1	I	Yes	BIGINT
NUMC	N	Yes	VARCHAR
CURR, QUAN, DEC, PREC	P	Yes – see below	DECIMAL, NUMERIC
TIMS	T	Yes	TIME

SAP Data Type Name	ABAP Type	Supported	TDV Data Type
LRAW, RAW	X	Yes – see below	VARCHAR

In addition to the columns, descriptive text is retrieved from SAP and stored in the Annotation field of TDV.

Several limitations occur if you introspect with RFC_READ_TABL:

- By default, columns greater than 512 bytes cannot be read, and are automatically ignored, when introspecting SAP tables.
- In SAP versions 5.0, and 6.0, accessing tables containing a floating point (FLTP) data type are not supported. These tables appear in the introspector but an exception is thrown when they are used. (Floating point values are supported if you use the YRFC_READ_TABLE.)
- RAW and LRAW columns (ABAP Type X) are not encoded properly, and may appear truncated when selected.
- Tables containing columns of STRING or RAWSTRING data type cannot be accessed. These tables appear in the introspector, but an exception is thrown when they are used.
- Some data in decimal columns (known as P (ABAP), Packed, BCD (Binary Coded Decimal), and DEC) appears as DECIMAL with native type Px.y, where x is the number of digits and y is the number of decimal places. DECIMAL(9,5) truncates positive numbers greater than 999.99999 and negative numbers less than -99.99999 because they exceed the nine characters allocated for the field. (YRFC_READ_TABLE can resolve this limitation.)

  To avoid returning silently incorrect data, the SAP Adapter raises an error when it encounters rows that contain truncated values.

TDV recommends using YRFC_READ_TABLE for your SAP data source whenever possible.

SAP note 758278 provides an implementation of YRFC_READ_TABLE that addresses decimal issues. To apply this fix, install the YRFC_READ_TABLE function module on SAP, then modify the following three advanced data source properties:

- Table read function: YRFC_READ_TABLE
- Table row length: 4010
- Table decimal fix: Checked

Capabilities and SAP Tables

Capabilities determine how a SQL statement is divided for execution among SAP, TDV, and other data sources that may be referenced in the query. Capabilities are a system of classifying SQL elements to account for the unique features and limitations of data sources. Where a capability is lacking in a data source but required to run a query, TDV performs the work itself.

The following table lists commonly used capabilities and how they apply to SAP tables. limitations For efficient queries, minimize use of non-push capabilities.

Capability	Supported in SAP?	Pushed to SAP?	Notes
Filter	Yes	Yes	Filter comparing two columns of same table is not allowed by SAP.
Filter–BETWEEN	Yes	Yes	
Filter–IN	Yes	Yes	
Filter-LIKE	Yes	Yes	
Functions–CAST	Yes	Yes	
CASE	Yes	No	
DISTINCT	Yes	No	
Functions–aggregate	Yes	No	
Functions–others	Yes	No	
GROUP BY	Yes	No	
Join	Yes	No	
ORDER BY	Yes	No	
Subquery	Yes	No	
UNION	Yes	No	
DELETE	No	No	SAP tables are read-only.
INSERT	No	No	SAP tables are read-only.

Capability	Supported in SAP?	Pushed to SAP?	Notes
Transactions	No	No	INSERT/UPDATE/DELETE not supported.
UPDATE	No	No	SAP tables are read-only.

Joins and SAP Tables

This section describes how to join SAP tables with each other and with other resources. Joins cannot be pushed to SAP; instead, every row of the joined tables must be fetched (a procedure known as a table scan). The technology used to communicate to SAP from TDV is not optimized for working with large data sets, so table scans should be avoided.

A semijoin is the best way to reduce the number of SAP rows that TDV retrieves and processes. To force a semijoin to occur in a query, add it immediately before the tables to be joined. For example:

```
SELECT * FROM A INNER { option semijoin } JOIN B ON A.K = B.K
```

Values of A.K are collected and passed in a query to B as the filter.

```
SELECT * FROM A
SELECT * FROM B WHERE K IN ({values of A.K from previous query})
```

Many rows in A can result in lengthy queries against B. If queries against B are large, TDV automatically breaks them into smaller queries, executes them separately, and reassembles a unified result set.

Put the larger table on the right side of the join whenever possible. When running a new query for the first time, activate the Execution Plan tab in Studio and click Execute and Show Statistics. Examine each node's row count and query after processing has begun, to make sure filters are pushed down to SAP.

If the interaction between TDV and SAP is still unclear and performance is poor, enable debug logging as described in the *TDV Installation and Upgrade Guide*.

TDV SQL Support for SAP ABAP Queries

The following sections describes how SAP ABAP queries work within TDV:

- Introspection and ABAP Queries, page 82
- Capabilities and ABAP Queries, page 83

Introspection and ABAP Queries

Introspection examines SAP for its list of ABAP queries and InfoSets, allows the user to select them, and translates their metadata to relational tables that TDV understands. The resource hierarchy and metadata mapping of ABAP queries is explained here.

- Resource Hierarchy, page 82
- Metadata Mapping, page 82

Resource Hierarchy

The SAP Adapter provides ABAP Queries and InfoSets as resources in TDV. In general, any ABAP query that can be executed via transaction SQ01 and output using the SAP List Viewer can be accessed through TDV.

ABAP queries are introspected by expanding the ABAP Queries folder within an SAP data source. The first level of folders separates ABAP queries into Global (cross-client) or Standard (client-specific). The second level of folders separates ABAP queries by their functional areas.

Infosets are introspected by expanding the InfoSets folder within an SAP data source. In general, any InfoSet in transaction SQ01 that can be used in an InfoSet query can be accessed through TDV.

Infosets are introspected by expanding the InfoSets folder within an SAP data source. The first level of folders separates InfoSets into Global (cross-client) or Standard (client-specific).

Metadata Mapping

Translating metadata from ABAP queries to TDV ABAP query resources is straightforward. Each output field of the ABAP query becomes a column in TDV with the same name. Selection fields in the ABAP query also become columns in TDV, but their name is prefaced with an underscore character to indicate their role as specifiers of filter constraints to SAP.

Each data type in SAP is mapped to a TDV SQL-based type. The following table lists the SAP data types found in ABAP queries, whether they are supported in TDV, and their corresponding TDV type.

SQL Data Type Name	ABAP Type	Supported in TDV?	TDV Data Type
CHAR, UNIT, CUKY, CLNT, LANG, LCHR, VARC	C	Yes	VARCHAR
DATS	D	Yes	DATE
FLTP	F	Yes	DOUBLE
STRING, XSTRING	g, y	No – not supported by ABAP queries	
INT4, INT2, INT1	I	Yes	BIGINT
NUMC	N	Yes	VARCHAR
CURR, QUAN, DEC, PREC	P	Yes	DECIMAL, NUMERIC
TIMS	T	Yes	TIME
LRAW, RAW	X	No – not supported by ABAP queries	

Capabilities and ABAP Queries

Capabilities determine how a SQL statement will be divided between SAP, TDV, and other data sources that may be referenced in the query.

The following table lists commonly used capabilities and how they apply to SAP ABAP queries. Supported indicates whether or not the SQL capability is supported in queries against ABAP queries. Pushed indicates whether the capability is supported directly on SAP. For efficient queries, minimize use of non-push capabilities.

Capability	Supported?	Pushed?	Notes
CASE	Yes	No	

Capability	Supported?	Pushed?	Notes
DELETE	No	No	ABAP queries and InfoSets are read-only.
DISTINCT	Yes	No	
Filter	Yes	Yes	
Filter–BETWEEN	Yes	Yes	
Filter–IN	Yes	Yes	
Filter–LIKE	Yes	No	
Functions–aggregate	Yes	No	
Functions–CAST	Yes	Yes	
Functions–others	Yes	No	
GROUP BY	Yes	No	
INSERT	No	No	ABAP queries and InfoSets are read-only.
Join	Yes	No	
ORDER BY	Yes	No	
Subquery	Yes	No	
Transactions	No	No	ABAP queries and InfoSets are read-only.
UNION	Yes	No	
UPDATE	No	No	ABAP queries and InfoSets are read-only.

TDV SQL Support for SAP Dates

By default, the SAP application data service throws an exception and aborts a query if erroneous dates are returned from the SAP system.

To override SAP date behavior

1. Uncomment the following line in the product.properties file:

```
sap.handle.erroneous.date=USE_DEFAULT
sap.handle.erroneous.date.default=99991231
```

2. To enable the properties, edit product.properties at:

```
<TDV _install_dir>\apps\dlm\app_ds_sap\conf\product.properties
```

3. The date format is YYYYMMDD. If no default date is specified, the date will be 19000101. If the specified default date is invalid, an exception is generated.

Security for SAP with TDV

This topic describes TDV support for SAP security features. It assumes knowledge of SAP's security infrastructure.

- Required Authorizations, page 85
- Troubleshooting Security-Related Errors, page 86
- Custom Security on SAP Tables, page 87

Required Authorizations

The following authorizations are required:

- To log in to SAP from TDV and introspect BAPIs and RFCs, the following authorization object is required:

  Class: AAAB (Cross-application Authorization Objects, Object: S_RFC (Authorization Check for RFC Access

  This object controls access to RFCs by function group. The most restricted configuration of S_RFC is to allow access to only the function groups required by TDV, then add new groups as necessary.

- To log in to SAP from TDV, the following function groups are required:

  SYST (System interface)

  SYSU (RFC resource administration)

  SDIFRUNTIME (Interfaces for Type Runtime Objects)

  RFC1 (RFC utilities)

- To introspect SAP metadata, the following function groups are required:

  SEM5 (Generic Browser for ABAP/BOR classes)

SWOR (Runtime System)

SDTX (Desktop Access)

- To introspect and query SAP ABAP queries and InfoSets, the following function groups are required, in addition the authorizations above:

  AQCF (SAP Query: Catalog functions)

  AQRC (SAP Query: Remote query call)

- To introspect and query SAP tables, in addition the authorizations above, the following authorization object is required:

  Class: BC_A (Basis: Administration), Object: S_TABU_DIS (Table Maintenance)

Troubleshooting Security-Related Errors

Some of the common security-related errors and their remedies are described below.

- RFC Authorization, page 86
- QUERY_TABLE Authorization, page 86

RFC Authorization

ERROR: User TESTUSER1 has no RFC authorization for function group SYST.

TDV is logging into SAP with a SAP user that lacks the authorization object S_RFC. Set authorization S_RFC_ALL to grant access to all RFCs.

S_RFC grants the user permission to introspect and invoke functions. But functions may still fail unless the appropriate authorizations for its parent application are granted. Depending on the function, application-level authorization errors may be reported in a return code. For example, invoke BAPI_COMPANYCODE_GETLIST without the proper application-level authorization and the field RETURN MESSAGE contains:

You do not have authorization to display company codes.

QUERY_TABLE Authorization

ERROR: com.sap.mw.jco.JCO$AbapException: (126) NOT_AUTHORIZED: User not authorized to access QUERY_TABLE.

TDV is logging into SAP with a SAP user that lacks the authorization object S_TABU_DIS. Set authorization S_TABU_ALL to allow display of all table classes, or limit access to specific groups of tables using the appropriate authorizations.

Custom Security on SAP Tables

The granularity of access control provided by S_TABU_DIS may be too coarse for some security requirements. For example, SAP administrators may prefer to allow access table by table to TDV users. In this case the authorization check on S_TABU_DIS in RFC_READ_TABLE is insufficient.

TDV supports the ability to call a custom implementation of RFC_READ_TABLE. This is accomplished by editing the data source to set the Table read function and Table row length to match the function name and field size of the custom function. The column names and data types in the custom function must be identical to those in RFC_READ_TABLE.

Pooled and Cluster Tables

This section describes TDV SQL support for SAP pooled and cluster tables.

- Understanding Pooled and Cluster Tables, page 87
- HR Clusters, page 89

Understanding Pooled and Cluster Tables

SAP uses three approaches to table storage:

- Transparent tables are stored as regular tables in the back-end database.
- Pooled tables are several logical tables stored as a single table, or "table pool," in the database.
- Cluster tables are logical tables stored as a single table, or "table cluster," in the database.

The difference between table pools and table clusters is that the component tables of a table cluster share common keys, while component tables of a table pool do not.

The TDV can access all three types of SAP tables and all views based on them.

Table Type	Table Class	Supported?	Examples
Transparent	TRANSP	Yes	KNA1, AFKO
Cluster	CLUSTER	Yes	BSEG, BSET
Pooled	POOL	Yes	A001, FINK
View	VIEW	Yes	V_T001, CUSPROJ

Some tables in SAP have misleading names. For example, “HR Cluster 1” (PCL1) is actually a transparent table. To ascertain the storage type of a table, go to the ABAP Dictionary (transaction code SE11), provide the name of the table, and display its technical settings.

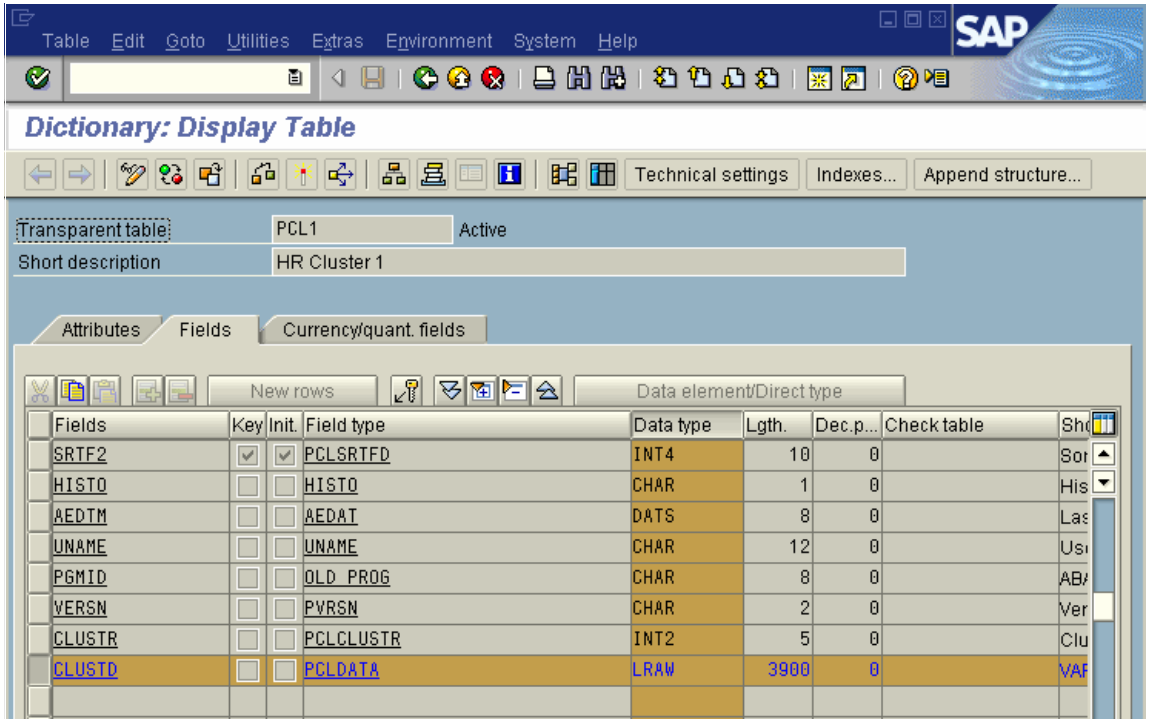
Working with the definitions provided above, TDV is able to access all three types of tables: transparent, pooled, and cluster. They all look to TDV like any other table resource. However, pool and cluster tables are more likely to contain columns that TDV cannot read:

Types LRAW and RAW—Due to limitations of the SAP function RFC_READ_TABLE, which TDV uses to access SAP tables, the column types RAW and LRAW are not properly handled.

Columns greater than 512 bytes—RFC_READ_TABLE does not support any type of columns that are greater than 512 bytes. Such columns are ignored during introspection, and a WARNING message is added to the log.

HR Clusters

HR functional areas contain several transparent tables with columns of type RAW and LRAW. Many have “cluster” in their names even though they are not cluster tables. “HR Cluster 1” (table PCL1) is one example. The inaccessible CLUSTD column is highlighted below.



Because SAP is unable to provide TDV access to RAW columns or columns of this length, administrators must write custom BAPIs to expose their data. TDV can then access the custom functions like any other resource.

SAP Global Properties

Global properties apply to all instances of an SAP data source. You can edit the properties to control how TDV works with SAP. Global properties are stored in the file:

<TDV_install_dir>\apps\dlm\app_ds_sap\conf\product.properties

Edit the properties using a text editor such as Notepad on Windows. TDV must be restarted for property changes to take effect.

The global properties are described below in alphabetical order.

sap.annotations.enabled

Default Value: true

Description:

If set to false, introspection of functions (BAPIs and RFCs) will not include their annotations.

sap.aq.introspectDbAccesses

Default Value: 1

Description:

ABAP Query database access limit to be used during introspection.

During introspection of ABAP queries, each introspected query is executed with no parameters to obtain a list of its output fields. (If this is not possible, other metadata query methods are used). Because the goal is to gather metadata only, the number of database accesses should be set to the smallest value that still results in at least one row.

If the database access limit is set too low, no data is selected from the ABAP query and the fallback method of metadata access is used, which may not include some calculated fields.

If problems are encountered with ABAP query metadata, test the ABAP query using transaction SQ01 until it returns rows, then increase this property value to match.

sap.debug

Default Value: false

Description:

If set to true, all data returned from SAP requests is logged to CSV files in the SAP driver folder. Each request becomes a separate CSV file. This can be helpful in debugging joins and performing data validation.

sap.feature.key

Default Value: BRT

Description:

Do not change this. TDV uses it internally for SAP certification.

sap.handle.erroneous.date

Default Value: USE_DEFAULT

Description:

Commented out by default. If enabled, specifies how to handle erroneous dates returned from the SAP server. SE_DEFAULT tells the connector not to throw an exception, but instead to use the default date specified by sap.handle.erroneous.date.default, page 91 (the SAP default property for handling erroneous dates).

sap.handle.erroneous.date.default

Default Value: 99991231

Description:

Commented out by default. If enabled, specifies the default date to use if an erroneous date is returned by the SAP server.

sap.introspect.tableFolders

Default Value: T5,T7,V_,/B,/BI,/BIC,/BI0,/BIC/,/BI0/,/BIC/B,/BIC/D,/BI0/D

Description:

Comma-separated list of table folder name patterns to contain subfolders.

During introspection, SAP tables are split by default into two levels of folders by name. For example, folder A contains AA, AB, AC, and so forth.

Some table name patterns may contain so many tables that SAP throws an exception when opening them. This property allows such name patterns to be expanded into an additional level of subfolders. For example, expanding T5 results in further folders T5A, T5B, etc.

sap.jco.traceLevel

Default Value: 2

Description:

Turns on the SAP JCo trace. Allowed levels are 0 through 10. The commonly used ones are:

0 - nothing

1 - errors

2 - errors and warnings

3 - info messages, errors and warnings

4 - execution path, info messages, errors and warnings

5 - verbose execution path, info messages, errors and warnings

6 - verbose execution path, limited data dumps, info messages, errors and warnings

7 - full execution path, data dumps with metadata, verbose info messages, errors and warnings

8 - full execution path, full data dumps with metadata, verbose info messages, errors and warnings

sap.table.decimalFix

Default Value: false

Description:

If true, SAP Note 758278 or equivalent has been applied to SAP so that the table read function encodes decimal values properly.

sap.table.functionName

Default Value: RFC_READ_TABLE

Description:

An SAP function to be used for accessing table resources.

sap.table.maxOptions

Default Value: 500

Description:

This property should never be changed.

sap.table.memoryAvailable

Default Value: 50

Description:

Memory available for SAP table queries, in megabytes. This is a critical property for performance tuning of SAP table access.

This property determines how many table rows can be requested from SAP in one round-trip, and thus how many round-trips to SAP are required to satisfy a user query. Each round-trip can only fetch the permitted number of rows, based on this memory constraint.

To calculate the number of rows per request, take available memory after applying sap.table.requestFactor and divide by two times the table row length. To examine requests and the number of rows in each, enable SAP driver. The ROWCOUNT parameter is the number of rows TDV is requesting from SAP.

If this property is set too high, the JVM hosting TDV can terminate abruptly when SAP's client connectivity library (JCo) attempts to allocate too much memory from its native code. The main purpose of this property is to guard TDV against these low memory conditions.

If this property is set too low, queries are split into many small requests to SAP, resulting in poor performance. With many concurrent requests, the amount of memory available may fall so low that the rows requested fall below the property sap.table.minimumSize. In that case, queries will block until requests are finished and memory is freed. If memory is still not available after a period of time (property sap.table.memoryMaxWait), the query is terminated with an exception.

The maximum heap size of the JVM (-Xmx setting) must be considered when changing this property value. If this value is increased without a corresponding reduction in the Java heap size, JVM crashes could result.

For example, a machine with 2 GB of RAM, a Java heap size of 600 MB permits a sap.table.memoryAvailable setting of up to 200 MB. Reducing the Java heap size to 500 MB enables an increase to 300 MB.

sap.table.memoryMaxWait

Default Value: 60

Description:

Amount of time to wait (in seconds) for memory to become available for an SAP request. If there is insufficient memory available for greater than this time period, an exception is thrown and the query is terminated.

The exception text is:

```
Could not reserve memory for request {0}: lowerBound = {1},
upper bound = {2}, reserved = {3}, total {4}.
```

The five variables in the message (in order from 0 to 4) are: number of concurrent requests, minimum size of query (bytes), maximum query size

(bytes), amount of memory reserved (bytes) globally for SAP table access, and amount of memory available (bytes) globally for SAP table access.

If queries are being terminated with this error message, decrease Java heap size and increase sap.table.memoryAvailable, or optimize SAP queries to avoid table scans.

sap.table.minimumSize

Default Value: 500

Description:

Minimum number of rows allowed for SAP table requests. When the system is in a low memory state, requests become fragmented, resulting in more round-trips to SAP to fetch data. Splitting a query into too many small requests typically degrades performance more than waiting for enough available memory to make fewer, larger requests.

This property forces queries to block until other queries release memory. If a query blocks for longer than sap.table.memoryMaxWait (seconds), an exception is thrown.

sap.table.profiler.enabled

Default Value: false

Description:

TDV does not use this.

sap.table.requestFactor

Default Value: 0.8, 0.25, 0.15, 0.1

Description:

Comma-separated list of values used to determine how much memory is available for reading SAP tables based on the number of concurrent requests. The number of concurrent requests is used as an index into this list. If this number is greater than the number of values, the last value is used.

For example (using the default value): The first request receives 90% of the available memory. If a second request is made before the first completes, it receives 50% of the remaining memory.

sap.table.rowLength

Default Value: 512

Description:

Length of data field returned by table read function, in bytes.

sap.table.safetyFactor

Default Value: 0.9

Description:

TDV does not use this.

sap.use.logon.lang.introspection

Default Value: false

Description:

If false, the SAP connection uses the language configured in the data source. If true, the SAP connection uses the language set in the SAP connection attributes, which originate from SAP's configuration of the SAP user.

Using SAP BW with TDV

This topic describes important usage information for SAP BW data sources.

About SAP BW Data Sources

Requirements

- You need an SAP BW account with API-level access enabled for your organization to work with this data source type.
- For installation requirements, see the *TDV Installation and Upgrade Guide*.

Large data sources with complex schemas and big tables with lots of columns have a correspondingly large amount of metadata that can pose special challenges for TDV. For TDV, it is not the amount of data stored, but rather the complexity and amount of metadata that defines the data source that is the most important factor for performance, particularly during introspection. For additional information on how to use TDV when introspecting large data source, see the *TDV User Guide*.

Considerations for introspection:

- Selecting a parent node selects all of the children resources in that node.
- Resources are organized by InfoArea as in BEx Query Analyzer. InfoProviders are further organized into three subfolders: Hierarchies, InfoProviders, and Queries.

- Introspecting resources is time-intensive, so it is best to select only the resources necessary for your project. If you find you need additional resources later, you can always add them through the Add/Remove Resources in Studio.
- Avoid using the Find field during introspection of SAP BW data sources. TDV cannot load all of the SAPBW metadata before introspection. Attempts to use the introspection Find feature can result in SAP BW database service interruptions.

Application Views for SAP

The organization of the views and folders mirrors that of SAP's Business Object Repository (BOR). Field names are aliased to provide human-readable names. In some cases several objects in BOR are synthesized in a single query to produce a more detailed, unified view of the data.

A common pattern employed in application views is that of List and Details. Views with names ending in "List" return a minimal set of columns that serve to uniquely identify an object. The identifiers are passed as arguments into a corresponding view, its name ending in "Details," to produce a more extensive set of columns.

Filter Data from Application Views

There are two ways to filter data from SAP using application views: using a filter provided by SAP itself, or by filtering within TDV. Both methods are used by application views depending on the view and the capabilities of SAP.

SAP BW Basic Tab

Field	Description
Application Server	Name of the machine hosting SAP BW or the host machine's IP address. For load-balanced configurations of SAP BW, leave this property empty.
SAP Router String	Routing entry. If an SAP Router is used to connect to the Application Server, enter its routing entry here. For example: /H/saprouter/H/194.117.106.130/S/3297/H/
System Number	Two-digit gateway service number of the SAP BW system.

Field	Description
Client	Three-digit client number of the SAP BW system.
User and Password	Valid user name and password to SAP BW.
Save Password check box	This option works in combination with the Pass-through Login option. By default, this option is disabled and un-editable. It becomes editable when you select the Pass-through Login option. If you accept the default, the password is saved and the Pass-through Login option remains disabled. Then, you can perform the following operations without having to supply the password again: • Introspect the current data source. • Reintrospect the data source. • Add/Remove data source resources. • Perform query/update/insert operations on a table in the data source. • Invoke a stored procedure. Refresh a cached view based on the data source resources.
Pass-through Login	Works in combination with the Save Password option. By default, this mode is Disabled and the password is saved. This mode is referred to as non-pass-through mode. Refer to the details given for the Save Password option. If you select the Enabled option, the Save Password option becomes editable. This mode is referred to as pass-through mode.

The operations you can and cannot perform in **pass-through mode** depend on if Save Password is checked as follows:

Save password?	Operations you can perform	Operations you cannot perform
Yes	Introspection. You do not have to resupply the password.	N/A

Save password?	Operations you can perform	Operations you cannot perform
No	• Query/update/insert/delete operations. You need to resupply the original login credentials for the current session. • Reintrospection, Add/Remove data source resources. You will be prompted to resupply the password that was used when the data source was originally introspected.	Schedule reintrospection.

SAP BW Advanced Tab

Some properties refer directly to the configuration of the SAP BW Server and must be provided by an SAP BW Administrator. Other properties are specific to TDV and how it interacts with SAP BW.

Option	Description
Maximum connections in pool	SAP JCo parameter specifying the maximum number of simultaneous connections in an SAP BW connection pool.
Maximum idle connections in pool	SAP JCo parameter specifying the maximum number of simultaneous idle connections, by destination, to keep open in an SAP connection pool. Set to 0 to disable connection pooling, closing connections after each request.
Maximum rows	Maximum number of rows to return from an SAP BW query. Default: 250,000. This setting prevents the return of datasets that are too large. To return all rows from the query, set this property to zero. This maximum row count is cumulative for all nodes, so queries or data sources using the “leaf nodes only” mode might receive significantly fewer rows than the maximum rows setting. This setting only works on rows being returned from DataStore Objects.
Show leaf nodes only check box	Determine how TDV handles hierarchical data from SAP BW. If unchecked, all nodes of the data hierarchy are returned. If checked, only members of dimensions which do not contain children are returned, and summary data is suppressed. Default: checked.

Option	Description
Include children of selected members check box	Display the children of the introspected components in the resource tree.
Allow large (>1M cells) datasets check box	Lets you introspect large datasets (those with more than 1M cells). Default: unchecked.
Display Descriptive Names	Display only descriptive names in the resource tree and in the Introspection dialog box. (Technical names still appear in the popup when the cursor hovers over the resource name, and on the resource's Info tab.) Use this radio button consistently for any given SAP BW data source; otherwise, results can be unpredictable.
Display Technical and Descriptive Names	Display both technical names and descriptive names in the resource tree and in the Introspection dialog box.
Display Technical and Descriptive Names, Use Technical Names In TDV Resource Namespace	Display technical names in the resource tree, and both the technical names and the descriptive names in the Introspection dialog box.

Reintrospecting SAP BW

Because each SAP BW data source contains resources that are unique to it, it is best to follow a few special steps when you reintrospect to make sure that you can see all expected metadata within Studio.

If you removed any SAP BW resources from the Studio view of the SAP BW data source metadata, reintrospection restores those resources to the Studio resource tree.

Note: If you reintrospect an SAP BW data source, keep the descriptive/technical name choice originally used for that data source. If you change it, results are unpredictable.

To reintrospect SAP BW

1. From Studio, open the SAP BW data source.
2. Click ADD/Remove Resources.
3. Select one or more schemas that you want to introspect or reintrospect.

   Avoid using the Find field during introspection of SAP BW data sources. TDV cannot load all of the SAPBW metadata before introspection. Attempts to use the introspection Find feature can result in SAP BW database service interruptions.
4. Select the following check boxes:
   - Re-Introspect previously introspected resources
   - Allow partial introspection, omitting resources with errors
   - Detect New Resources During Re-Introspection
5. Click **Next**.
6. Review the summary.
7. Click **Finish** to begin reintrospection.
8. When the status message indicates that reintrospection was successful or completed, click **OK** to close the dialog.

Introspecting Large Datasets with SAP BW

Large data sources with complex schemas, big tables, lots of columns, or many dimensions have a correspondingly large amount of metadata that can pose special challenges for TDV. For TDV, it is not the amount of data stored, but rather the complexity and amount of metadata that defines the data source that is the most important factor for performance, particularly during introspection.

To work with large datasets and prevent runaway MDX queries when using SAP BW within TDV

1. When defining your SAP BW data source, on the Advanced tab, set the Allow large (>1M cells) datasets check box. For other data source definition information, see About SAP BW Data Sources, page 97.

   This specifies that large datasets that are greater than 1M cells can be introspected. By default, datasets that are larger than one million cells cannot be introspected.

2. Define an OLAP View, following the instructions in Creating a View from a Cube, page 103.
3. For your OLAP View, generate the Execution Plan. See Generating an OLAP View Execution Plan, page 107.
4. Review the Estimated Rows Returned value that is displayed for the Execution Plan.
5. Review the other information displayed in the execution plan. If TDV estimates that your query could return more than one million cells of data from the SAP BW database, the FETCH node of the Cardinality Statistics panel displays a message indicating this and suggesting that you define some constraints on the query to limit the data set to return.
6. Determine if limiting the number of dimensions used for the OLAP View could improve retrieval of the dataset. See Creating an OLAP View, page 104.
7. Determine if adding filter conditions would help improve performance. See Creating Filter Conditions on an OLAP View, page 105.
8. Save your work and run another Execution Plan for comparison.
9. Set up and use a test system to determine if the performance of the OLAP View is acceptable.

Creating a View from a Cube

Studio provides a way to create views from multidimensional cubes. The Studio OLAP View editor lets you graphically design and display cube slices. Studio introspects and displays the available dimensions, measures, hierarchies, and members of a selected cube to allow graphical selection of the parameters that create basic SQL queries. Basic queries are converted into MDX to retrieve cube data into a table cursor for viewing and use.

After you have the OLAP cube data in a simplified table view, the data can be processed further by procedure, by triggers, or by inclusion of the OLAP View into other views to process or cache data that might be difficult to handle or slow to retrieve using direct queries to the SAP BW source.

The Info tab of an OLAP View reveals the full and formal resource name, type, owner, and lock information.

This topic includes:

- Using the OLAP View Editor SQL Tab, page 107

Creating an OLAP View

The Studio OLAP View editor lets you graphically design and display cube slices. This editor effectively flattens a cube into a two-dimensional object that can be used by relational database tools.

To create an OLAP view

1. Make sure you have the SAP BW Data Adapter installed following the instructions in the TDV Installation and Upgrade Guide.
2. Create and introspect the SAP BW data source.
3. In the Studio resource tree, select a location, right-click, and choose New OLAP View.
4. Open an SAP BW data source and select the cube on which to base the view.
5. Name your new cube view and click **OK** to proceed.

   Note: Uninitialized or unconfigured SAP BW cube resources return detailed error messages that might report various BAPI misconfiguration states. Studio introspection of misconfigured SAP BW cubes passes these data source configuration errors to the Studio user for review. The error messages should be forwarded to the SAP BW resource owner. Further configuration of the SAP BW cube might be necessary if that resource is to be used.

   The OLAP View editor is opened and displayed in Studio. The TDV-defined resource name of the cube that you selected appears next to the Associated OLAP Cube label.

   The Select tab displays available dimensions and measures of the associated OLAP view.
6. Select the Available Dimensions that you want in your flattened view and use the arrows to move them into the Selected Dimensions field.

   Use restraint in the selection of dimensions. Every dimension in the SELECT is cross-joined, making it possible to return millions of cells and cause excessive memory usage in shared libraries. For example, SELECT * is not recommended on SAP BW resources.
7. Select the Available Measures that you want in your flattened view and use the arrows to move them into the Selected Measures field.

   Both dimensions and rows appear as table columns, with dimensions displayed, followed by measures.

8. Execute the query created by your selections by clicking the Execute button. The first 50 rows of results appear.
9. (Optional) Change the order in which the selected dimensions and measures are requested from the data source using the up- and down-arrows.

   The results of query execution shown in the Result tab persist even after other changes are made to selections.
10. Execute the query again to see the most recent changes.
11. Save the view to make changes permanent.

Creating Filter Conditions on an OLAP View

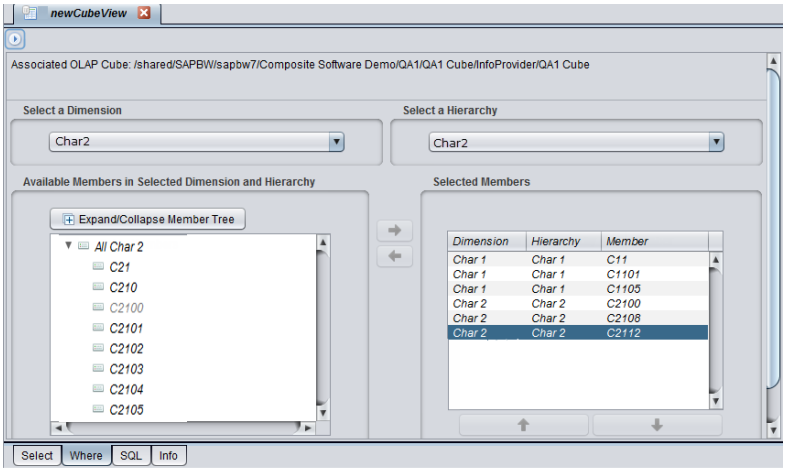
Filter conditions for an OLAP view created from an SAP BW cube can be set using the Where tab that is part of the OLAP View editor. You can use filters to restrict the result set to cross-linked dimensional rows that match the selected dimension hierarchy members.

Member selection creates a WHERE filter condition, and all returned rows must have members equal to the selections. All additional selected members add a filter by means of an AND in the WHERE clause. All rows returned by multiple selected member filters must satisfy all filter conditions.

To set filter conditions on an OLAP view

1. Open a view that you have created from an SAP BW cube.
2. Select the Where tab.

The Where tab displays the available dimensions in the associated OLAP view.



3. Select a dimension and/or a hierarchy from the list.

   In SAP BW cube views, selecting a dimension displays its hierarchy levels.

4. Use the Expand/Collapse Member Tree button to query the cube for available members.

   For very large cubes, the round-trip to the data source can take more than a minute.

5. Select a member by clicking it and moving it to the Selected Members field.

   Member selection creates a WHERE filter condition, and all returned rows must have members equal to the selections. All additional selected members add a filter by means of an AND in the WHERE clause. All rows returned by multiple selected member filters must satisfy all filter conditions.

6. Save the OLAP View.

   If the resource indicates it is impacted, some unconfigured state or error has occurred, in which case you should check your View.

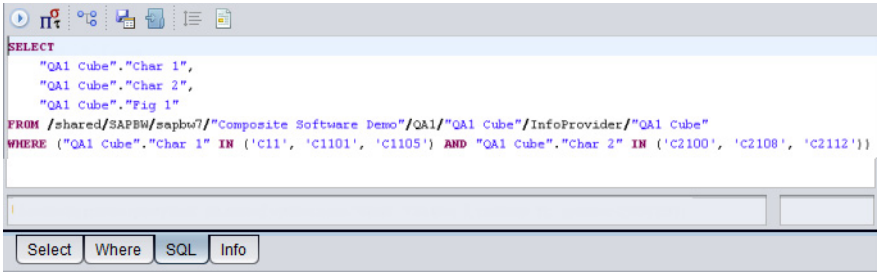
   Selecting dimensions, measures, hierarchies, and members immediately generates the SQL associated with such selections.

Using the OLAP View Editor SQL Tab

The SQL tab of the OLAP View editor displays the SQL that will be run against the SAP BW data source to retrieve data. You can use the tab to review how selections on the other editor tabs have been interpreted and you can use the tab to edit the SQL query directly.

To use the SQL tab

1. Open a view that you have created from an SAP BW cube.
2. Select the SQL tab.



3. Inspect the SQL generated by your selections at any time by opening the SQL tab.
4. Edit the SQL displayed in the SQL tab.

   Note: Directly editing the SQL disables the Select and Where tabs of the OLAP View editor.

Generating an OLAP View Execution Plan

Execution plans are important tools when designing resources to be consumed by others, especially if performance is a high priority.

To generate and OLAP View Execution Plan

1. Open a view that you have created from an SAP BW cube.
2. Select the SQL tab.
3. Click Show Execution Plan on the editor toolbar.
4. Review the Execution Plan Node and Field Contents. Descriptions for how to interpret much of this information can be found in the *TDV User Guide*.

5. Review the Estimated Rows Returned value.

Accessing Data Lineage for an OLAP View

Data lineage can be use to determine where metadata originates from. This can help you track how changes to certain data sources can have effects on data that clients are consuming.

To generate and OLAP View Execution Plan

1. Open a view that you have created from an SAP BW cube.
2. Select the SQL tab.
3. Click the Show Lineage Panel toolbar button on the editor toolbar.

   The Lineage panel opens in the lower section of the view's editor and displays all the resources involved with the view in a graphical format. Descriptions for how to interpret much of this information can be found in the *TDV User Guide*.

SQL Support for SAP BW

This section describes SQL support for SAP BW and its resource types—functions, tables, and ABAP queries.

Understanding SAP BW with TDV

TDV has made the interface between TDV and SAP BW as efficient and accurate as possible. However, due to some product limitations and proprietary interfaces, TDV and SAP BW results might differ, and performance issues might occur when accessing SAP BW data using TDV. The following sections provide additional information.

Differences in Queries and Results

Queries in SAP BW using BEx are created differently in TDV. In TDV, SQL queries generate MDX. The process is different in SAP BW using BEx. This disparity can cause differences in results. Workarounds are described in this section. See Introspection Resources, page 110 for information about how TDV interprets SAP BW resources in TDV.

Queries with Input Values

Queries can require input values. To handle queries with input values in TDV, set the values using a WHERE clause and provide values for columns that begin with an underscore (_). For an example, see Variables, page 122.

Queries with Filters

BEx queries that contain filters might not return the same results in TDV as they do in BEx. To apply the filters in TDV, recreate them using a WHERE clause. This limitation is imposed by SAP BW; TDV does not have access to the filters.

Additional Columns in Query Results

BEx queries do not return the same columns in TDV as they do in BEx. For example, you might see 1,000 columns in TDV listing every property of every dimension in the query, even if you did not include those properties in BEx. This is because the SAP BW API does not give TDV access to all of the items selected for output in the query. TDV only gets a list of InfoObjects. From there, TDV expands each one to include all of its columns.

Performance Considerations

Performance of TDV's SAP BW Adapter is slower and more resource-intensive on the SAP BW server than accessing SAP BW through native SAP tools, because SAP does not give TDV access to proprietary APIs.

When performance issues occur when using TDV with SAP BW, you can reproduce them outside of TDV by copying the MDX statements generated by TDV (by examining the TDV cs_server.log) and entering them into the MDX test environment (transaction MDXTEST in SAP GUI). This allows SAP BW administrators to optimize settings and improve performance of the MDX statements.

Introspection Resources

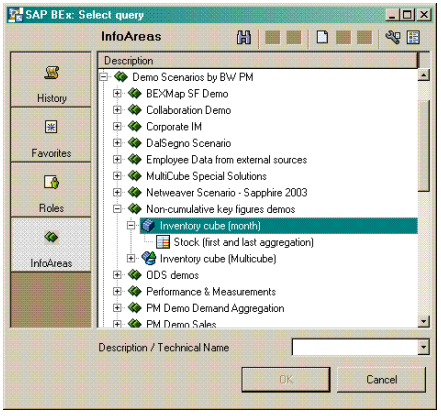
This section describes the details of introspecting SAP BW resources.

- Resource Hierarchy in SAP BW, page 110
- Resource Hierarchy in TDV, page 111
- Metadata Mapping, Multi-Dimensional, page 112
- Metadata Mapping, ODS, page 115

Resource Hierarchy in SAP BW

The organization of SAP BW resources in TDV mirrors SAP BW as much as possible. InfoAreas in SAP BW become folders and subfolders in TDV. Supported InfoObjects are organized into subfolders.

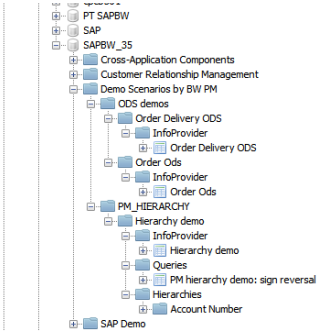
In the following screenshot of BeX Analyzer, nodes have been expanded in InfoAreas to reveal the InfoProvider Inventory cube (month) which contains the Query Stock (first and last aggregation).



The next section shows how the equivalent resources look in TDV.

Resource Hierarchy in TDV

As with BEx Analyzer, TDV displays only the objects that the user is authorized to see.



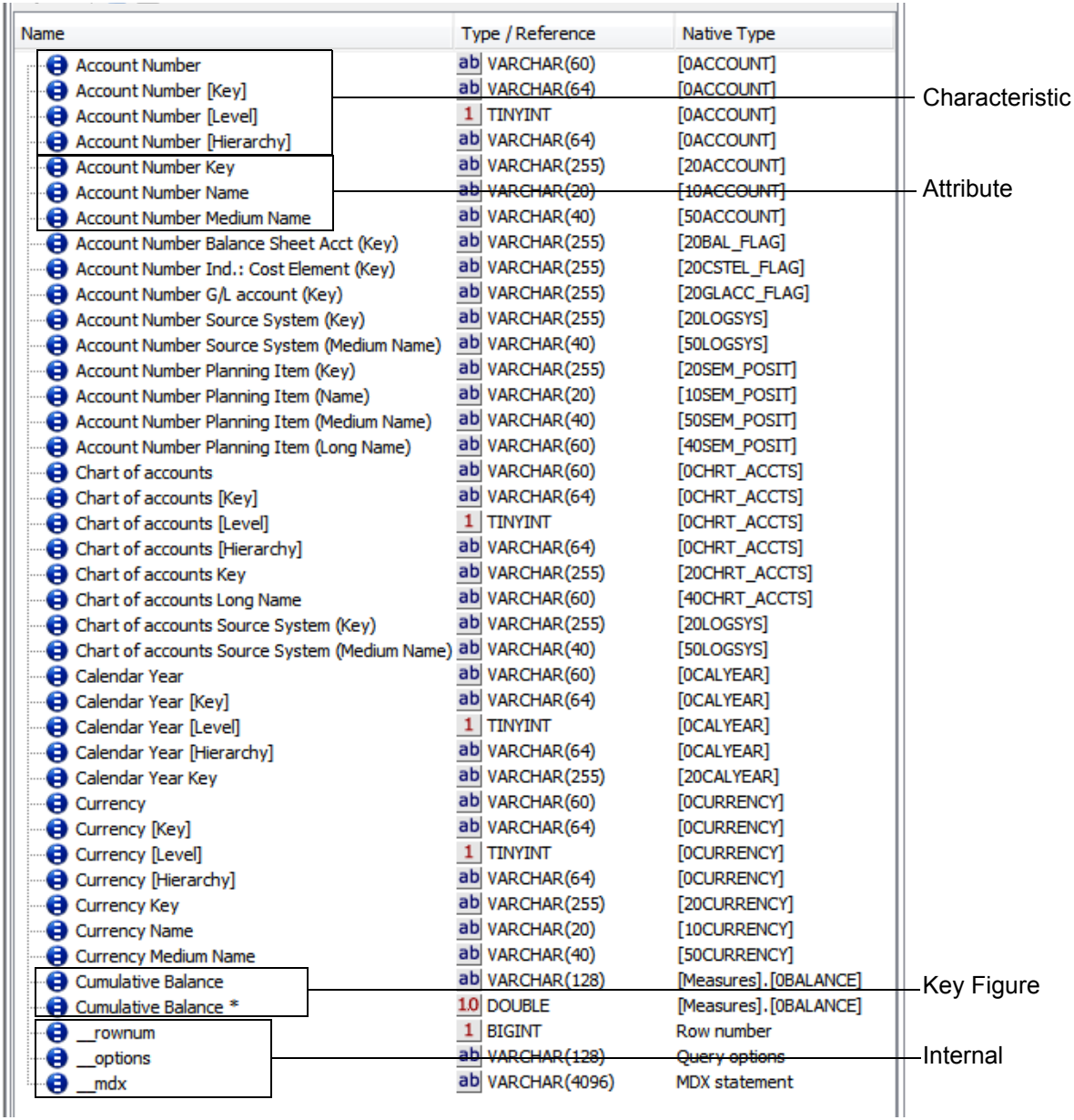
InfoAreas in BeX Analyzer are displayed as folders in Studio, and these folders are named after the InfoArea's technical name or description. Moving the mouse over an item reveals a tool-tip with its technical name and type.

InfoProviders contain up to three subfolders to group resources by type, as described below.

- InfoProvider: This folder contains a single resource, the InfoProvider itself. This resource provides direct access to the InfoProvider without requiring construction of a Query.
- Queries: This folder contains all of the Query objects built from the InfoProvider. Query objects, also known as Query Cubes, are views into an InfoProvider created in BeX Analyzer.
- Hierarchies: This folder contains subfolders for each Basic Characteristic referenced within the InfoProvider. Each Characteristic contains at least one Hierarchy resource. ODS Objects do not contain the Hierarchies folder.

Metadata Mapping, Multi-Dimensional

The data structures present in multi-dimensional InfoProvider, Query, or Hierarchy objects are transformed into a flat namespace of columns in TDV. InfoProviders and Queries contain three types of data relevant to TDV: Characteristics, Attributes, and Key Figures. Queries may contain Variables as well. Hierarchies contain only Characteristics and Attributes. The next screen shot illustrates the column types in a typical Query.



Characteristics

Each Characteristic is represented by four columns containing Characteristic metadata, differentiated by naming conventions:

- Characteristic—This column's name is just the Characteristic's short description. It contains the member caption. The native type is the technical name of the Characteristic.
- Characteristic [Key]—The key column contains the member name, without square brackets. The native type is the technical name of the Characteristic.
- Characteristic [Level]—The level column contains the level of the current member, where zero is the root of the hierarchy. The native type is the technical name of the Characteristic.
- Characteristic [Hierarchy]—The hierarchy column contains the hierarchy in which the current member belongs (64 character VARCHAR). The native type is the technical name of the Characteristic.

Attributes

Following the Characteristic columns are Characteristic Attributes. Each Attribute is named using the Characteristic short description, a space, and the Attribute description. Its native type is the technical name of the Attribute. Its data type is VARCHAR, with the length determined by SAP BW metadata, defaulting to 255 characters if metadata is not available.

Key Figures

Each Key Figure is represented as two columns. One column contains the formatted value; the other contains the unformatted, numeric value. Both columns are named using the Key Figure name, but the numeric value field has an asterisk (*) suffix to differentiate it. The native type is the technical name of the Key Figure. The data type of the formatted Key Figure is a 128 character VARCHAR. The data type of the unformatted value is DOUBLE.

Variables

Variables only appear in Query objects. They can be identified by the single underscore before their names. The data type is VARCHAR of length determined by SAP BW metadata. The native type is the technical name of the variable.

Internal Columns

Internal columns are created for use within TDV SQL and do not correspond to objects in SAP BW. Their names are prefaced with two underscores, and they appear at the end of the list of columns. Each Internal column has a unique function, as discussed in Query Mapping, page 118.

Metadata Mapping, ODS

The InfoObjects in an ODS InfoProvider are transformed into a set of columns in TDV.

Each InfoObject becomes a column, named using the InfoObject's short text. The native type is a concatenation of the InfoObject type and its SAP data type, for example "CHA C" is a Characteristic with SAP data type C. InfoObject types are listed below:

CHA—Characteristic

DPA—Data packet characteristic

KYF—Key figure

TIM—Time characteristic

UNI—Unit of measurement

The data type of the column is determined by its SAP data type. The supported SAP data types and their corresponding TDV data types are:

C—VARCHAR

D—DATE

F—DOUBLE

g—VARCHAR

I—BIGINT

N—VARCHAR

P—DECIMAL, NUMERIC

T—TIME

X—VARCHAR

Capabilities

Capabilities characterize the features and limitations of data sources. For example, an Oracle data source can execute subqueries, while SAP BW cannot. Capabilities are consulted when a query is processed so that data sources receive only the query processing work they support; otherwise, TDV performs the work itself.

This section includes the following topics:

MDX Capabilities

The following table lists commonly used capabilities and whether they are supported in queries against SAP BW. Pushed indicates whether query processing can be passed to SAP BW. For efficient queries, minimize use of non-push capabilities.

Capability	Supported	Pushed	Notes
Filter	Yes, with special usage	Yes	Filters are mapped to MDX as specified in Query Mapping, page 118.
Filter–IN	Yes, with special behavior	Yes	See Query Mapping, page 118
Functions–CAST	Yes	Yes	
ORDER BY	Yes	Yes	Filters are mapped to MDX as specified in Query Mapping, page 118.
CASE	Yes	No	
DISTINCT	Yes	No	
Filter–BETWEEN	Yes, with special usage.	No	Filters are mapped to MDX as specified in Query Mapping, page 118
Functions–aggregate	Yes	No	
Functions–others	Yes	No	
GROUP BY	Yes	No	
Join	Yes	No	See Joins and Query Performance , page 128.
Subquery	Yes	No	
UNION	Yes	No	
DELETE	No	No	
Filter-LIKE	No	No	
INSERT	No	No	
Transactions	No	No	

Capability	Supported	Pushed	Notes
UPDATE	No	No	

ODS Capabilities

The following table lists capabilities that apply to ODS Objects, whether they are supported against ODS Objects, and whether query processing can be pushed to SAP BW. For efficient queries, minimize use of non-push capabilities. Capabilities for ODS objects are:

Capability	Supported	Pushed	Notes
CASE	Yes	No	
DELETE	No	No	
DISTINCT	Yes	No	
Filter	Yes	Yes	All conditional operators are supported. The keywords OR, AND, and grouping of terms with parentheses have no impact on the query other than to indicate the list of filters. Filters containing NULL literals are ignored.
Filter—BETWEEN	Yes	Yes	
Filter—IN	Yes	Yes	See notes on “Filter—BETWEEN” above.
Filter—LIKE	Yes	No	
Functions—aggregate	Yes	No	
Functions—CAST	Yes	Yes	
Functions—others	Yes	No	
GROUP BY	Yes	No	
INSERT	No	No	
Join	Yes	No	See Joins and Query Performance , page 128.

Capability	Supported	Pushed	Notes
ORDER BY	Yes	Yes	Order is always ascending due to limitations of the SAP BAPI. ASC and DESC keywords are ignored.
Subquery	Yes	No	
Transactions	No	No	
UNION	Yes	No	
UPDATE	No	No	

Query Mapping

This section describes how queries are mapped from SQL to MDX. This mapping is critical to content and performance.

- Characteristics, page 118
- Attributes, page 120
- Key Figures, page 120
- Ordering, page 121
- Slicers, page 121
- Variables, page 122
- Custom MDX, page 123
- Query Options, page 123
- Row Number, page 124
- Time-Dependent Queries, page 125
- Enabling Tracing, page 125
- Working with Large Batch Sizes, page 126

Characteristics

Characteristics selected in SQL are projected onto the Query Axis ROWS (Axis 1). Multiple Characteristics are cross-joined, resulting in a Cartesian product of all selected Characteristics. This creates a rows-and-columns data set for TDV. Selecting many Characteristics could result in long-running queries and huge result sets.

A Characteristic projected without a filter adds its MEMBERS set to the cross-join. If the active Hierarchy contains more than two levels, the MDX DRILLDOWNLEVEL function is applied to produce a set containing all nodes of the Hierarchy. Additionally, the MDX HIERARCHIZE function is applied to order the members in their hierarchical order.

To restrict the Characteristic to specific members, apply a filter (WHERE clause) to the Characteristic or the Characteristic [Key] column. For example, the following query produces a cross-join of Company code, Division, and Sales organization, and restricts Sales organization to the one with key 2200:

```
SELECT
"Company code", "Division", "Sales organization", "Costs in
Document currency (SAP Demo)", "Billed Quantity (SAP Demo)"
FROM
"SAP DemoCube"
WHERE
"Sales organization [Key]" = '2200'
```

If a member key cannot be found in the Hierarchy, a warning is logged but the query continues. The log is in the file cs_server.log, located in the subdirectory logs of <TDV_install_dir>.

If you do not know the member key, you can use the member text, also known as the “caption,” but note that member texts may not be unique. To avoid ambiguous queries, it is best to filter on member keys.

The following query accomplishes the same effect as the query above, but refers to Sales organization 2200 by its text designation (Paris):

```
SELECT
"Company code", "Division", "Sales organization", "Costs in
Document currency (SAP Demo)", "Billed Quantity (SAP Demo)"
FROM
"SAP DemoCube"
WHERE
"Sales organization" = 'Paris'
```

Characteristics have the following functionality:

- All conditional operators and the BETWEEN keyword are supported. Operators other than equals (=) and not equals (<>) have special meaning for ranges, discussed next.
- To create a range from the natural order of the hierarchy, use the greater than (>) or less than (<) symbol.
- Ranges can be specified only on the Characteristic [Key] column.
- The keywords OR, AND, IN, and grouping of terms with parentheses have no impact on the query other than to indicate the list of members.

- Members can be excluded by using the not equals (<>) operator or the NOT keyword. These map to the EXCEPT set expression.
- If no members can be found for a member text, an exception is thrown.
- Filters are not allowed on the column Characteristic [Level] and cause an exception.
- ies are supported. To specify a non-default Hierarchy, filter on the Characteristic [Hierarchy] column. All operators are treated as equals (=). The filter value is a Hierarchy key (enclosed in square brackets) or name. Values that cannot be resolved to a Hierarchy, and multiple Hierarchy filters on the same Characteristic, cause an exception.

Attributes

Attributes, also known as Dimension Properties, typically contain data related to the Characteristic. For example, the Characteristic "Company Code" contains an attribute named "Country (Name)."

Filter behavior for Attributes is similar to that of ordinary SQL queries:

- All conditional operators are supported, plus the BETWEEN keyword.
- The keywords OR, AND, and NOT, and grouping of terms using parentheses, are supported.
- The IN keyword is supported.
- IS NULL is mapped to test for empty strings.

Unlike in MDX, Attributes can be selected without their parent Characteristic. TDV selects the parent Characteristic automatically. This adds a Characteristic to the cross-join, which increases the load on SAP BW, the network, and TDV.

Key Figures

Key Figures, also known as Measures, contain numeric data that is aggregated based on the sets present in the query. They are projected on the Query Axis COLUMNS (Axis 0).

Both the formatted and unformatted (numeric) values are available as columns and behave identically in queries,.

Filter behavior for Key Figures is similar to that of ordinary SQL queries. The following filters are supported:

- NULL is treated as empty string. IS NULL is mapped to test for empty string.
- All conditional operators are supported, as well as the BETWEEN keyword.

- The keywords OR, AND, and NOT are supported. Grouping of terms with parentheses is also supported.
- The IN keyword is supported.

Ordering

Ordering of data is accomplished with the ORDER BY clause, adapted to work with SAP BW in the following ways:

- Ordering on Characteristic, Attribute, and Key Figure is supported.
- Ordering by more than one column is not allowed and causes an exception.
- The sort keywords ASC and DESC are supported, within the hierarchy. To sort regardless of hierarchy, specify the orderBreaksHierarchy query option. For more information, see the section Query Options, page 123.
- Ordering on Internal, Characteristic [Level] or Characteristic [Hierarchy] columns cannot be pushed to SAP BW. To order on these columns, create a Parameterized Query and a View that selects from it.

Slicers

When a filter is placed on a column that is not projected, TDV treats the filter as a Slicer, placing it on the Slicer Axis. A Slicer qualifies tuples for each cell of the query, but does not affect the Query Axis. For example:

```
SELECT
Company Code, Division, Sales
FROM
DemoCube
WHERE
Calendar Month/Year = 'JAN 2001'
```

The above query returns the Sales Key Figure for the cross-join of Company Code and Division Characteristics, with cells restricted to tuples including the January 2001 member of the Calendar Month/Year Characteristic. This may result in many empty rows where Company Code/Division contains no Sales data, but these are removed by default. (To display empty rows, specify the query option includeEmptyRows. For more information, see the Query Options, page 123.)

Slicers have the following behavior:

- The only column types permitted as Slicers are Characteristic and Characteristic [Key]. Others are ignored.
- All conditional operators except not equals (<>) are treated as equals (=).
- Multiple members are combined into a set. The keywords OR and AND are treated identically as means of delineating members belonging to the set.

- Members can be excluded by using the not equals (<>) operator or the NOT keyword. These map to the EXCEPT set expression.
- To specify a range from the natural order of the hierarchy, use greater than (>) or less than (<).
- Ranges without an upper and lower bound cause an exception.
- Only the Characteristic [Key] column type can be used to specify ranges. Ranges on other columns cause an exception.
- If the Slicer Axis consists of more than one member, it may be necessary to specify the order of members in the MDX. This is not possible from SQL, as the TDV query engine does not preserve the order of terms in a WHERE clause. To control the query to this extent, use custom MDX. See Custom MDX, page 123 for further information.

Note: In BeX Query Designer, Slicers are called Filters.

Variables

Variables are features of SAP BW Queries. To provide a Variable value, add a filter to the SQL statement. Depending on the variable type, the value can be a member key (enclosed in square brackets), member text, or literal.

In the following example, the Variable Sales organization is assigned to members whose text matches the values USA Philadelphia or USA Denver.

```
SELECT
"Business Partner", "Calendar Year/Month",
"Returns Quantity", "Returns Value"
FROM
"Returns per Business Partner"
WHERE
"_Sales organization" IN ('USA Philadelphia', 'USA Denver')
```

The annotation of a Variable column includes guidance for how to use it within a query.

Rules for using Variables in queries are as follows:

- Variables with selection type Complex are allowed to contain multiple values.
- Variables with selection type Value are only allowed to contain a single value.
- All conditional operators and the BETWEEN keyword are supported.
- The keywords OR, AND, IN, and grouping of terms with parentheses ,are supported, but have no impact on the query other than to indicate the list of variable assignments.
- Multiple values for the same variable are mapped to INCLUDING or, with the keyword NOT, EXCLUDING.

- NULL is treated as empty string. IS NULL is mapped to test for empty string.

Custom MDX

The Internal column "__mdx" in InfoProvider and Query resources allows custom MDX to be sent directly to SAP BW, bypassing SQL-to-MDX query mapping in TDV. This helps when you need to access features of MDX without using SQL; but then MDX queries cannot be used with other queries. TDV has no visibility to tune them.

The following is an example of using custom MDX in a query. The MDX function TOPCOUNT is used to provide ranking and ordering which cannot be accomplished in SQL:

```
SELECT
"Sales Personnel", "Calendar Year/Month",
"Net value of the invoice item in the docCurrency (SAP Demo)"
FROM
"SAP DemoCube"
WHERE
"__mdx" =
'SELECT NON EMPTY { [Measures].[0D_NETVLINV] } ON AXIS(0),
NON EMPTY {TOPCOUNT({[0D_SALE_EMP].[LEVEL01].MEMBERS *
[0CALMONTH].[LEVEL01].MEMBERS }, 3,
[Measures].[0D_NETVLINV])} ON AXIS(1)
FROM [$0D_DECU]
WHERE [0CALYEAR].[1998]'
```

Developing MDX can be challenging. It is best to work within a specialized OLAP tool to develop the initial query, test it, and then paste it into Studio.

Additionally, a number of rules must be followed for the MDX to execute properly within TDV:

- Axis 0 must contain Key Figures only.
- Axis 1 must contain Characteristics only.
- No other axes are permitted.
- The SELECT clause in TDV must contain corresponding columns for the Key Figures, Characteristics, and Attributes projected in the MDX statement. If any columns are missing or if any extraneous columns are introduced, the query fails, returns no data, or returns incomplete data.

Query Options

An internal column named "__options" (two underscores) is provided for every InfoProvider and Query resource. This column accepts a comma-delimited list of query options, which can modify the behavior of the SAP BW query in ways that cannot be expressed in SQL.

Valid options are the following:

- leafNodes: This option overrides the selection of the Data Source advanced property named Show leaf nodes only for the duration of the query. With this option, only members of dimensions that do not contain children are returned.
- allNodes: This option overrides the selection of the advanced property named Show leaf nodes only for the duration of the query. All nodes are returned exactly as received from SAP BW, regardless of their position in the hierarchy.
- orderBreaksHierarchy: When ordering rows with ORDER BY, the default behavior is to order members first by their position in the hierarchy, and then by the Characteristic, Attribute, or Key Figure specified in the ORDER BY clause. This option indicates that the hierarchy should be broken when ordering data.
- keyDate: Used for time-dependent queries. For more information see Time-Dependent Queries, page 125.
- includeEmptyRows: This option specifies that the NON EMPTY clause is to be omitted from the ROWS axis for this query. Depending on the SAP BW data and query, this may result in empty values in Characteristic or Attribute columns. By default, empty rows are removed. Here is an example of the includeEmptyRows option:

```
SELECT
   "Company code",
   "Division",
    "Sales organization",
    "Costs in Document currency (SAP Demo)",
   "Billed Quantity (SAP Demo)",
   "__rownum"
FROM
/shared/QA_SAPBW/Sources/SAPBW_35/"SAP Demo"/"SAP Demo Sales and
Distribution"/"SAP DemoCube"/InfoProvider/"SAP DemoCube"
WHERE "Sales organization" = 'Paris'
AND "__options" = 'includeEmptyRows'
```

- includeEmptyColumns: Specifies that the NON EMPTY clause is omitted from the COLUMNS axis for this query. This may result in empty values in Key Figure columns. By default, empty columns are removed.

Row Number

The natural order of data from SAP BW may be important to preserve. Without an explicit ORDER BY clause, results from SAP BW are ordered hierarchically. Once the data is reordered due to a join or other SQL operation, it cannot be restored through ORDER BY or any other operator in TDV because it relies on the hierarchical order available only to SAP BW.

For this reason, an Internal column named "__rownum" (two underscores) exists for every SAP BW resource in TDV. Values in this column range from 0 to n, indicating the natural order of data as returned from SAP BW. This makes it possible to preserve the natural order of the data. This column can be selected, but not in a filter expression.

Time-Dependent Queries

Hierarchy nodes and other data in SAP BW can change. By default, queries run using the current date. To set a different date, create a filter on the "__options" Internal column, using the keyDate option and a date in YYYY-MM-DD format.

In the following example, the query is executed with a key date of January 1, 2006:

```
SELECT
"Sold-to party", "Net value of the invoice item in the docCurrency
(SAP Demo)"
FROM
SAP_DemoCube
WHERE
"__options" = 'keyDate 2006-01-01'
```

Enabling Tracing

Tracing logs every interaction with SAP BW to files so that data values can be examined and validated against output in TDV. Combined with the debug output in cs_server.log, this can be helpful for troubleshooting queries.

To enable tracing

1. Edit this properties file:

   <TDV install directory>\apps\dlm\app_ds_sapbw\conf\product.properties

2. Change the property sapbw.trace to true.
3. Restart the TDV Server.
4. Check the trace files that are stored in the following directory:
<TDV_install_dir>\apps\dlm\app_ds_sapbw

Note: Tracing affects performance and creates large files on disk, so be sure to change the sapbw.trace property to false and restart TDV Server when tracing is no longer needed.

Working with Large Batch Sizes

If your developers work with queries that return large amounts of data, or depend on the amount of available memory, you might be able to improve performance by adjusting the batch size.

Typically, the batch size should not be changed. If the setting is too high, the server requires too much memory to process the results from SAP BW and can crash. If the setting is too low, query performance can suffer because the server might unnecessarily split up single queries into multiple queries. However, depending on other factors such a network latency and bandwidth, it is possible that increasing the batch size could result in improved performance on queries that return amounts of data larger than the batch size.

- If the TDV server has a large amount of memory available to the JVM, the batch size might be increased.
- If the TDV server has a shortage of memory, the batch size might be decreased.

To enable working with large files

1. Open this properties file using your favorite text editor:

   ```
   <TDV_install_dir>\apps\dlm\app_ds_sapbw\conf\product.properties
   ```

2. Change the property sapbw.batchSize setting.

   By default, sapbw.batchSize is set to 25000 cells (TDV requests 25000 cells for each call to SAP BW). If there are more cells in the response, TDV makes more requests for 25000 cells until all of the data is transferred from SAP BW to TDV. The minimum setting for sapbw.batchSize is 1; the maximum is 999999.

3. Restart TDV Server.

Multidimensional Queries

Queries against SAP BW are multidimensional, which introduces a new behavior to TDV that may be unexpected to users accustomed to querying relational databases.

- The Show Contents Feature, page 127
- Create OLAP Views Using SAP BW Data Sources, page 127
- Work with Views, page 127
- Filter on Levels, page 127
- Clearing the SAP BW Metadata Cache, page 128

The Show Contents Feature

The feature Show Contents available in Studio is not recommended on any SAP BW resource. Show Contents performs a SELECT * query, which results in a cross-join of every possible Characteristic in the SAP BW resource.

Create OLAP Views Using SAP BW Data Sources

When creating a new view that uses an SAP BW data source in combination with other data sources, or when using multiple data sources of any type, use the New OLAP View command in Studio to create a new view.

If you are creating a view that calls only one SAP BW data source, you can create a New OLAP View in Studio. For more information about OLAP Views, see the *TDV User Guide*.

Work with Views

TDV Server makes a number of assumptions when optimizing queries, which can cause confusion. This is especially true when building Views that call other Views, or when using Views from outside of TDV using JDBC or ODBC. Because a cross-join is used to query SAP BW, when a Characteristic column is removed from a query due to optimization, the meaning of the underlying MDX query changes dramatically.

For example, if a View includes the Characteristics “Company Code” and “Division” and “Key Figure Sales,” an ODBC query selecting only “Division” and “Sales” produces a different data set than a query selecting all columns. To ensure the same base data are returned regardless of the query, wrap any Views using SAP BW in a Parameterized Query, and expose the Parameterized Query to query users. This prevents TDV from applying relational optimization rules to the multi-dimensional query.

Filter on Levels

You can filter on level columns (Dimension [Level]). For example, you can add a WHERE clause to the SQL statement that filters the [Level] field:

```
SELECT Dimension1, Dimension2, Measure
FROM Cube
WHERE "Dimension1 [Level]" = 2
```

This would return Measure for the cross product of Dimension1 (members beginning from level 2 and below) and Dimension2 (all members).

Clearing the SAP BW Metadata Cache

TDV caches SAP BW metadata to improve query performance. These caches are automatically cleared after every TDV restart, but you might want to do this at other times.

For example, your SAP BW data source might become out of sync with the SAP BW database, even after re-introspecting the data. If the metadata is still out of sync after re-introspection, you can restart the TDV instance, or you can clear the metadata cache as described in this section.

To clear the metadata cache

1. Open the SAP BW data source.
2. Select the Re-Introspection tab.
3. Click Clear Metadata Cache.

Joins and Query Performance

Joins cannot be pushed to SAP BW. Executing joins in TDV can degrade performance, because a table scan would be required, and it would fetch every row of the joined tables. The technology that TDV uses to connect with SAP BW is not optimized for large data sets, so table scans should be avoided.

A semijoin is the best way to reduce the number of SAP BW rows retrieved and processed. To force a semijoin to occur in a query, add the option immediately before the table to be joined. Example:

```
SELECT *
FROM X INNER {OPTION SEMIJOIN} JOIN Y ON X.Key = Y.Key
```

Values of X.Key are collected and passed in a query to Y as the filter:

```
SELECT * FROM X
SELECT * FROM Y WHERE Key IN ({values_of_X.Key_from_previous_query})
```

If X has many rows, queries against Y can be lengthy. If the queries against Y become too large, TDV automatically partitions them and reassembles the results into a unified set.

Put the table that returns the larger number of rows on the right side of the join whenever possible. When running a new query for the first time, activate the Execution Plan tab in the Studio and click Execute and Show Statistics. Examine each node's row count and query after processing has begun to make sure that filters are pushed down to SAP BW. This is a good way to see the mechanics of a semijoin in action. If the interaction between TDV Server and SAP BW is still unclear and performance is poor, enable debug logging for the adapter as described in the *TDV Installation and Upgrade Guide*.

Security for SAP BW with TDV

This topic describes TDV support for SAP BW security features. It assumes knowledge of SAP BW's security infrastructure.

Required Authorizations

The following authorizations are required to log into SAP BW from TDV and introspect Cubes and Queries::

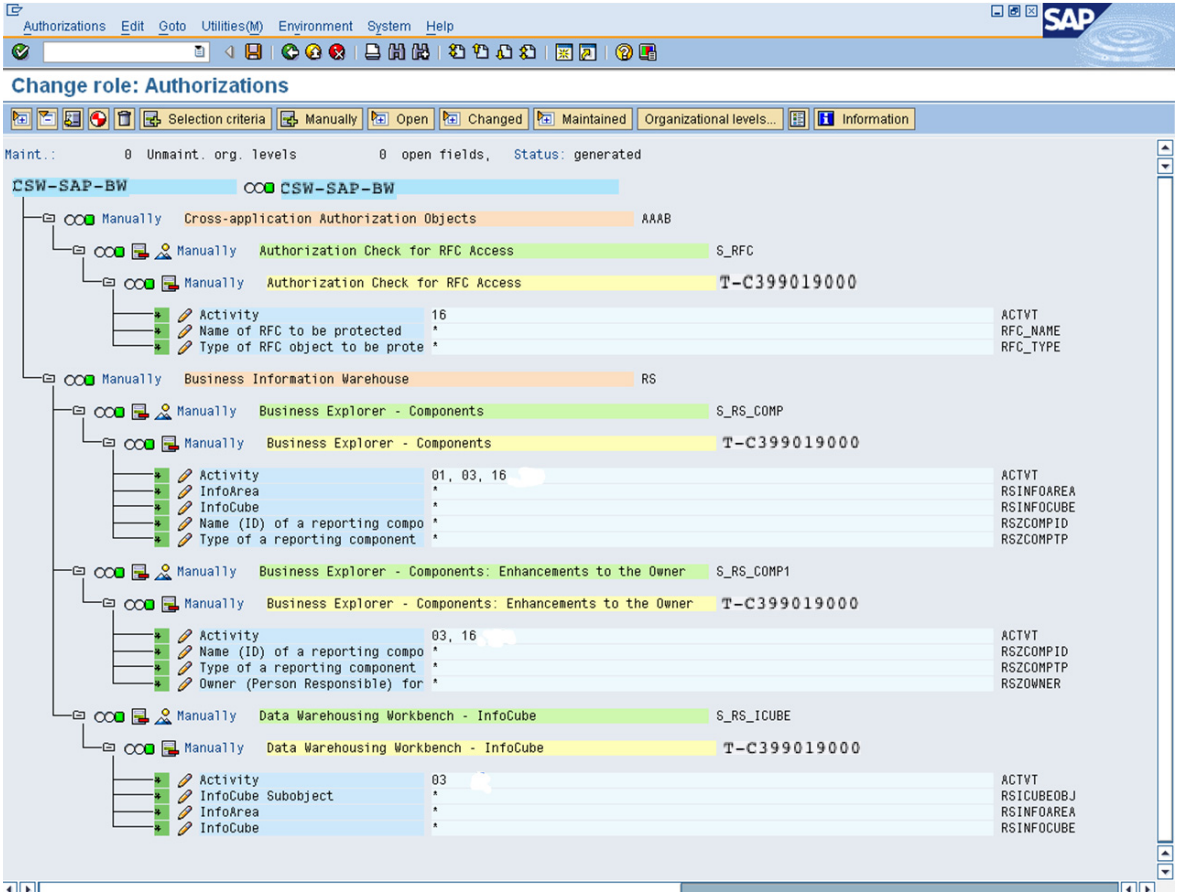
```
Class: AAAB (Cross-application Authorization Objects
Object: S_RFC (Authorization Check for RFC Access)
Field: Activity . Value : 16 ( Execute)
```

In addition, the following authorization objects can be used to limit the InfoAreas visible to the TDV user, as well as limiting the InfoProviders and Queries accessible:

```
Class: RS (Business Information Warehouse),
Object: S_RS_COMP (Business Explorer - Components)
Field: Activity . Value : 01 ( Create), 03 ( Display) and 16 (
Execute)
```

Note: The SAP ODBO interface does not modify InfoObjects but still Activity 01 is required. Otherwise SAP BW introspection does not fetch any metadata.

```
Class: RS (Business Information Warehouse),
Object: S_RS_COMP1 (Business Explorer - Components: Enhancements to
the Owner)
Field: Activity . Value : 03 ( Display) and 16 ( Execute)
Class: RS (Business Information Warehouse),
Object: S_RS_ICUBE (Administrator Workbench - InfoCube)
Field: Activity . Value : 03 ( Display)
```



Troubleshooting Security-Related Errors

This section describes common security-related errors and their remedies.

RFC Authorization

ERROR: User TESTUSER1 has no RFC authorization for function group SYST

TDV is logging into SAP with an SAP BW user that lacks the authorization object S_RFC. Set authorization S_RFC_ALL to grant access to all RFCs, or restrict them to only the OLAP BAPIs required by TDV:

BAPI_MDPROVIDER_GET_DIMENSIONS

BAPI_MDPROVIDER_GET_MEASURES

BAPI_MDPROVIDER_GET_PROPERTIES

BAPI_MDPROVIDER_GET_CATALOGS

BAPI_MDPROVIDER_GET_CUBES

BAPI_MDPROVIDER_GET_HIERARCHYS

BAPI_MDPROVIDER_GET_LEVELS

BAPI_MDPROVIDER_GET_MEMBERS

BAPI_MDPROVIDER_GET_VARIABLES

BAPI_MDDATASET_GET_CELL_DATA

BAPI_MDDATASET_DELETE_OBJECT

BAPI_MDDATASET_CREATE_OBJECT

BAPI_MDDATASET_SELECT_DATA

BAPI_MDPROVIDER_SET_KEY_DATE

BAPI_MDDATASET_GET_AXIS_DATA

BAPI_MDDATASET_GET_AXIS_INFO

SAP BW Global Properties

This topic describes the SAP BW global properties file that you can edit to control how TDV works with SAP BW. Global properties apply to all instances of the SAP BW data source. You can edit the properties to control how TDV works with SAP BW.

SAP BW Global Trace Level Property

Global properties are stored in the file:

<TDV _install_directory>\apps\dlm\app_ds_sapbw\conf\product.properties

Edit global properties using a text editor such as Notepad on Windows. TDV must be restarted for property changes to take effect.

There currently is only one global property for SAP BW.

sap.jco.traceLevel

Default Value: 2

Description: Turns on the SAP JCo trace. Allowed levels are zero through ten. The commonly used ones are:

0—Nothing

1—Errors

2—Errors and warnings

3—Info messages, errors and warnings

4—Execution path, info messages, errors and warnings

5—Verbose execution path, info messages, errors and warnings

6—Verbose execution path, limited data dumps, info messages, errors and warnings

7—Full execution path, data dumps with metadata, verbose info messages, errors and warnings

8—Full execution path, full data dumps with metadata, verbose info messages, errors and warnings

Using SAP BW BEx with TDV

This topic gives an overview of TDV adapters, and describes important usage information for SAP BW BEx data sources.

For installation requirements, see the *TDV Installation and Upgrade Guide*.

Query Limitations and Workarounds

The following limitations and workarounds apply to TDV when it is used with the BEx adapter:

- The adapter cannot perform aggregate operations.
- A query that uses the Conditions on Selected Characteristics feature of BEx returns an exception in Studio.
- If a query contains execution steps that cannot be pushed to the data source, the query as a whole is executed in TDV.
- In many cases you can express a BEx query as a parameterized query so that it always pushes its parameters.

Application Views for SAP

The organization of the views and folders mirrors that of SAP's Business Object Repository (BOR). Field names are aliased to provide human-readable names. In some cases several objects in BOR are synthesized in a single query to produce a more detailed, unified view of the data.

A common pattern employed in application views is that of List and Details. Views with names ending in "List" return a minimal set of columns that serve to uniquely identify an object. The identifiers are passed as arguments into a corresponding view, its name ending in "Details," to produce a more extensive set of columns.

Filter Data from Application Views

There are two ways to filter data from SAP using application views: using a filter provided by SAP itself, or by filtering within TDV. Both methods are used by application views depending on the view and the capabilities of SAP.

SAP BW BEx Characteristics

This section describes the characteristics of SAP BW BEx data sources:

Quantity of Metadata

Large data sources with complex schemas and big tables with many columns have correspondingly large amounts of metadata, which can have significant impact on TDV performance. For TDV, the most important factor affecting performance is the amount and complexity of metadata that defines the data source, particularly during introspection. For additional information on how to use TDV when introspecting large data sources, see the *TDV User Guide*.

TDV examines the TABLE-COLUMN relationships in the metadata and queries the data dictionary for full column names.

Introspection and Query Behavior

- Time-Dependent Hierarchies—No introspection or SQL changes are required to use time-dependent hierarchies. The adapter uses the default settings for each hierarchy in a query, such as its version and effective date.
- Multiple Structures in Columns Axis—TDV projects two or more structures onto the columns axis. Before TDV version 7.0, the adapter flattened multiple structures projected on the rows axis, but assumed a single structure (typically Key Figures) on the columns axis.

  TDV tables display technical names of the variable from BEx Query Designer. Users are only exposed to the final value in reports, calculated from an input (prompt) parameter. The correct data is returned, but query developers might prefer to alias the column name to a friendlier one in a TDV View.
- Maximum Cells—Maximum Cells is an internal proprietary value that is passed from the BEx Adapter directly to BW. It appears to limit the number of cells processed by BW in preparing a result, but in fact Maximum Cells controls the amount of resources that SAP BW uses for a query. Maximum Cells can be set to a value greater than its defaul value of 500,000.

Search Behavior

In the SAP BEx adapter's introspection UI, the Search box does not work the way it does for most other data sources.

The BEx adapter loads nodes on demand. Initially, if you search for a BEx query, you get no hits, because nothing has been navigated yet. All TDV knows at this point is the list of top container names.

To see queries, you must pick a top level container, navigate down to a folder containing BEx queries, and expand this folder. Those objects are now indexed and searchable.

To make other queries searchable, you need to go back to the top and drill down another folder path and expand *that* list of BEx queries. TDV can return BEx queries from all paths you have navigated from top level to query folders.

Back-end File Preparation

A back-end properties file (introspection.properties) containing customer-defined variable values is needed for introspection of BEx queries. If a BEx query cannot be introspected, try again after setting up this file with default values for any variables the query might need. Default values enable TDV to configure the BEx query's state correctly to obtain its metadata.

The location of this file is:

<TDV_install_dir>/apps/dlm/app_ds_sapbwbex/conf/

Each input parameter in this file has one line with the query's technical name, the input parameter name, and its literal value. For example, for a query named QRT7_D2_GMT_008 and its five variables, you would use:

```
QRT7_D2_GMT_008.QRT7ABC=NONE
QRT7_D2_GMT_008.QRT_D2_RUNIT=BG0007
QRT7_D2_GMT_008.QRT7DEFG=ISOLATED
QRT7_D2_GMT_008.QRT7HJK=STRICT
QRT7_D2_GMT_008.QRT7LMN=LOLD
```

The assigned values just need to be plausible default values. They do not need to be exact for the query to execute.

Those values are used even during normal queries, unless values are provided for required variables. The BEx query is in an invalid state and cannot be executed if required variable values are not provided.

The file is designed to provide values in "unattended mode" (that is, during introspection). In normal queries, you should explicitly provide values for your variables rather than relying on the introspection.properties file.

Maximum Cells

Maximum Cells is an internal parameter of the SAP BW RFC that TDV uses when retrieving cells from an open query. The BEx Adapter simply passes the value to SAP BW.

Maximum Cells limits the resources that SAP uses to execute a query and retrieve results. It is not an upper limit on the result set, like LIMIT in SQL.

BW Statistics Tables

The BEx Adapter calls an extra RFC to cause SAP to write statistics to the OLAP statistics table, mirroring the behavior of the BEx Analyzer. These statistics help the user to understand query performance characteristics before queries are allowed to run on production systems.

SAP BW BEx Basic Tab

When you create an SAP BEx data source, you need to set basic connection properties. For details, see the *TDV User Guide*.

Field	Description
Application Server	Name or IP address of the machine hosting SAP BW BEx. For load-balanced configurations of SAP BW BEx, leave this property empty.
SAP Router String	Routing entry. If an SAP Router is used to connect to the Application Server, specify its routing entry here. Sample: /H/saprouter/H/194.117.106.130/S/3297/H/
System Number	Two-digit system number of the SAP BW BEx instance. Also known as the gateway service number.
Client	Three-digit client number of the SAP BW BEx system.
User and Password	Valid user name and password to SAP BW BEx.
Save Password (check box)	This option works in combination with the Pass-through Login option. By default, this option is disabled and cannot be edited. It becomes editable when you select the Pass-through Login option. If you accept the default, the password is saved and the Pass-through Login option remains disabled. In this case, you can perform the following operations without having to supply the password again: • Introspect the current data source. • Reintrospect the data source. • Add or remove data source resources. • Perform query/update or insert operations on a table in the data source. • Invoke a stored procedure. • Refresh a cached view based on data source resources.
Pass-through Login	Works in combination with the Save Password check box (above). By default, this mode is disabled (non-pass-through mode).and the password is saved. If you select the Enabled option (pass-through mode), the Save Password check box is enabled.

The operations you can and cannot perform depend on whether or not Save Password is checked, as follows.

Save Password?	Operations You Can Perform	Operations You Cannot Perform
Yes	• Introspection. You do not need to resupply the password.	N/A
No	• QUERY. You need to resupply the original login credentials for the current session. • Reintrospect a data source, or add or remove data source resources. You are prompted to provide the same password that was used when the data source was originally introspected.	Schedule reintrospection.

SAP BW BEx Advanced Tab

Some of these properties refer directly to the configuration of the SAP BW BEx Server and must be provided by an SAP BW BEx Administrator. Other properties are specific to TDV and how it interacts with SAP BW BEx.

Option	Description
Maximum Connections in Pool	SAP JCo parameter specifying the maximum number of simultaneous connections in an SAP BW BEx connection pool.
Maximum Idle Connections in Pool	SAP JCo parameter specifying the maximum number of simultaneous idle connections (in an SAP connection pool) kept open by the destination. A value of 0 means no connection pooling (each connection closed after each request has completed).
Maximum Cells	Maximum Cells is an internal proprietary value that is passed from the BEx Adapter directly to BW. Maximum Cells controls the amount of resources that SAP BW uses for a query. Maximum Cells can be set above its default value of 500,000.
Display Descriptive Names	Display only descriptive names in the resource tree and in the Introspection dialog box. (Technical names still appear in the popup when the cursor hovers over the resource name, and on the resource's Info tab.) Use this set of three "display" radio buttons consistently for any given SAP BW data source; otherwise, results can be unpredictable.

Option	Description
Display Technical and Descriptive Names	Display both technical and descriptive names and use them in the TDV resource tree. Use this set of three “display” radio buttons consistently for any given SAP BW data source; otherwise, results can be unpredictable.
Display Technical and Descriptive Names, Use Technical Names In TDV Resource Namespace	Display technical names in the TDV resource tree (“namespace”), and both the technical names and the descriptive names in the Introspection dialog box. Use this set of three “display” radio buttons consistently for any given SAP BW data source; otherwise, results can be unpredictable.

Reintrospecting SAP BW BEx

Because SAP BW BEx contains resources that are unique to it, you need to take added steps when you reintrospect it, to make sure that you can see all expected metadata within Studio.

If you removed any SAP BW BEx resources from the Studio view of its metadata, the following steps make sure those resources are put back into the Studio resource tree.

To reintrospect SAP BW BEx

1. From Studio, open the SAP BW BEx data source.
2. Click Add/Remove Resources.
3. Select one or more schemas that you want to reintrospec

   Avoid using the Find field during introspection of SAP BW BEx data sources. TDV cannot load all of the SAP BW BEx metadata before introspection. Attempts to use the introspection Find feature can result in SAP BW BEx database service interruptions.
4. Select the following check boxes:
   - — Reintrospect previously introspected resources
   - — Allow partial introspection, omitting resources with errors
   - — Detect New Resources During Reintrospection
5. Click Next.

6. Review the summary.
7. Click Finish to begin reintrospection.
8. When the status message indicates that reintrospection was successful or completed, click OK to close the dialog box.

Clearing the Metadata Cache

TDV caches SAP BEx metadata to improve query performance. These caches are automatically cleared after every TDV restart, but you might want to do this at other times.

For example, your SAP BEx data source might become out of sync with the SAP BEx database, even after reintrospecting the data. If the metadata is still out of sync after reintrospection, you can restart the TDV instance, or you can clear the metadata cache as described in this section.

To clear the metadata cache

1. Open the SAP BEx data source.
2. Select the Re-Introspection tab.
3. Click Clear Metadata Cache.

Generating an Execution Plan

Execution plans are important tools when designing resources that are to be consumed by others, especially if performance is a high priority.

Note: Pay particular attention to the Estimated Rows Returned value, which might be very large.

To generate an execution plan

1. Open a view that you have created from an SAP BW BEx data source.
2. Select the SQL tab.
3. Click Show Execution Plan on the editor toolbar.
4. Review the Execution Plan Node and Field Contents. Descriptions for how to interpret much of this information can be found in the *TDV User Guide*.

Also see Query Limitations and Workarounds, page 133.

Accessing Data Lineage

You can use data lineage to determine where metadata originates, so you can track how changes to data sources can affect data that clients are consuming.

To view data lineage

1. Open a view that you have created from an SAP BW BEx.
2. Select the SQL tab.
3. Click the Show Lineage Panel toolbar button on the editor toolbar.

The Lineage panel opens in the lower section of the view's editor and displays all the resources involved with the view in a graphical format. Descriptions for how to interpret this information can be found in the *TDV User Guide*.

SQL Support for SAP BW BEx

This section describes SQL support for SAP BW BEx resource types and their behavior within TDV SQL queries.

- Introspection Resource Hierarchy, page 141
- SQL Capabilities, page 142

Introspection Resource Hierarchy

This section describes the details of introspecting SAP BW BEx resources, including how the resource hierarchy appears in TDV and how metadata is mapped in TDV.

The organization of SAP BW BEx resources in TDV mirrors SAP BW BEx as much as possible.

InfoAreas in BEx Analyzer are displayed as folders in TDV, and these folders are named after the InfoAreas' descriptions. If a description for an InfoArea is not available, its technical name is used in TDV. Move the mouse over an item to see a tooltip with its technical name and type.

SQL Capabilities

SQL capabilities characterize the features and limitations of data sources. For example, an Oracle data source can execute subqueries, while SAP BEx cannot. Capabilities are consulted when a query is processed so that data sources receive only the query processing work they support; otherwise, TDV performs the work itself.

The following table lists capabilities that apply to ODS Objects only—whether they are supported in queries against ODS Objects or can be pushed to SAP BEx for execution. For efficient queries, minimize use of non-push capabilities.

The capabilities for ODS objects are listed in the table.

Capability	Supported	Pushed	Notes
BETWEEN	Yes	Yes	See notes in WHERE, page 143.
CASE	Yes	No	
CAST	Yes	Yes	
DELETE	No	No	
DISTINCT	Yes	No	
Filters	Yes	Yes	See Filters, page 143.
Functions	Yes	No	CAST is pushed. Other functions are supported but not pushed.
GROUP BY	Yes	No	
IN	Yes	Yes	See notes in WHERE, page 143.
INSERT	No	No	
JOIN	Yes	No	See Joins and Semijoins, page 143.
LIKE	Yes	No	
ORDER BY	Yes	Yes	Because of SAP BAPI limitations, order is always ascending (ASC). The ASC and DESC keywords are ignored.
Subquery	Yes	No	
WHERE	Yes	Yes	See WHERE, page 143.

Filters

All conditional operators are supported. The keywords OR, AND, and grouping of terms with parentheses are supported, but have no impact on the query other than to indicate the list of filters. Filters containing NULL literals are ignored. Filters on key figures are not supported and are ignored.

Joins and Semijoins

Joins cannot be pushed to SAP BW. Meanwhile, executing joins in TDV can degrade performance, because a table scan would be required, and it would fetch every row of the joined tables. The technology that TDV uses to connect with SAP BW BEx is not optimized for large data sets, so it is best to avoid table scans.

A semijoin is the best way to reduce the number of SAP BW BEx rows retrieved and processed by TDV. To force a semijoin to occur in a query, add it immediately before the table to be joined. For example:

```
SELECT *
FROM X INNER {OPTION SEMIJOIN} JOIN Y ON X.Key = Y.Key
```

Values of X.Key are collected and passed in a query to Y as the filter:

```
SELECT * FROM X
SELECT * FROM Y WHERE Key IN ({values of X.Key from previous
query})
```

Note: TDV semijoins involving BEx queries echo the input values (underscore columns) in the result set.

If X has many rows, queries against Y can be lengthy. If the queries against Y become too large, TDV automatically partitions them and reassembles the results into a unified set.

Whenever possible, put the table that returns the larger number of rows on the right side of the join. When running a new query for the first time, activate the Execution Plan tab in the Studio and click Execute and Show Statistics. Examine each node's row count and query after processing has begun to make sure that filters are pushed down to SAP BW BEx. This is a good way to see the mechanics of a semijoin in action. If the interaction between TDV Server and SAP BW BEx is still unclear and performance is poor, enable debug logging as described in the *TDV Installation and Upgrade Guide*.

WHERE

SQL WHERE processing brings TDV closer to parity with BEx Analyzer and other tools in their ability to execute BEx queries. The BEx adapter supports these selection types:

- Single Value—Characteristic, Hierarchy, and Hierarchy Node.

- Several Single Values—Multiple parameter (or prompt) values of three variable types can appear in a single SQL statement via the IN clause. These variable types are: Characteristic, Hierarchy, and Hierarchy Node.
- Intervals—The BEx adapter can set upper and lower bounds of an intervals using:
  - A BETWEEN clause
  - An equivalent pair of inequality operators

Exclusion Lists—Using the SQL NOT operator or equivalent inequality operator, you can modify the selection type of a variable value to exclude rather than (by default) include. This applies to all three supported selection types (above).

Security for SAP BW BEx with TDV

This topic describes assumes knowledge of SAP BW BEx's security infrastructure.

- Required Authorizations, page 144
- Troubleshooting Security-Related Errors, page 145

Required Authorizations

The following authorizations are required to log into the SAP BW BEx data source from TDV and introspect queries:

```
Class: AAAB (Cross-application Authorization Objects)
Object: S_RFC (Authorization Check for RFC Access)
Field: Activity . Value : 16 ( Execute)
```

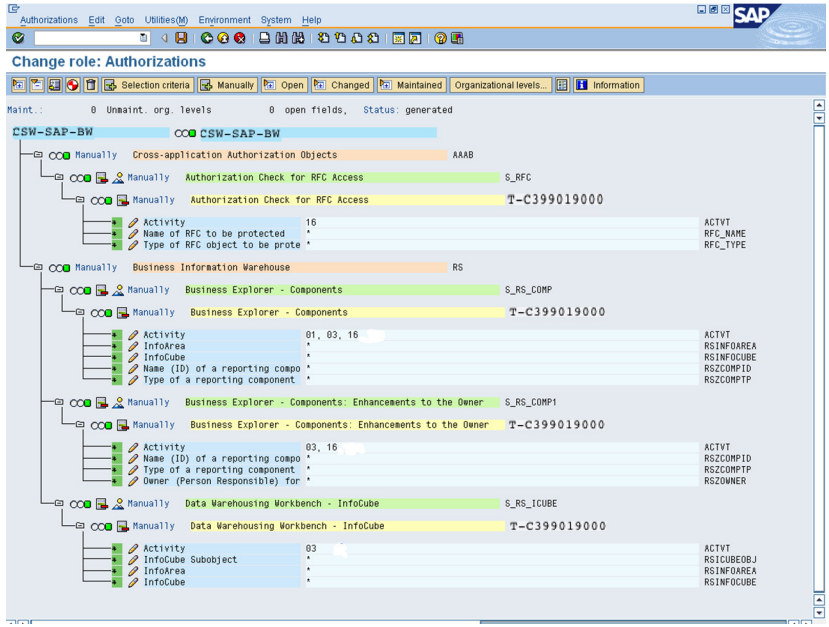
In addition, the following authorization objects can be used to limit the InfoAreas visible to the TDV user, as well as limiting the InfoProviders and Queries accessible:

```
Class: RS (Business Information Warehouse)
Object: S_RS_COMP (Business Explorer - Components)
Field: Activity . Value : 01 ( Create), 03 ( Display) and 16 (
Execute)
```

Note: Although the SAP ODBO interface does not modify InfoObjects, Activity 01 is still required. Otherwise, SAP BW BEx introspection does not fetch any metadata.

```
Class: RS (Business Information Warehouse),
Object: S_RS_COMP1 (Business Explorer - Components: Enhancements to
the Owner)
Field: Activity . Value : 03 ( Display) and 16 ( Execute)
```

```
Class: RS (Business Information Warehouse),
Object: S_RS_ICUBE (Administrator Workbench - InfoCube)
Field: Activity . Value : 03 ( Display)
```



Troubleshooting Security-Related Errors

This section describes common security-related errors and their remedies.

RFC Authorization

ERROR: User TESTUSER1 has no RFC authorization for function group SYST

TDV is logging into SAP with an SAP BW BEx user that lacks the authorization object S_RFC. Set authorization S_RFC_ALL to grant access to all RFCs, or restrict them to only the OLAP BAPIs required by TDV:

BAPI_MDPROVIDER_GET_DIMENSIONS	BAPI_MDPROVIDER_GET_MEASURES
BAPI_MDPROVIDER_GET_PROPERTIES	BAPI_MDPROVIDER_GET_CATALOGS
BAPI_MDPROVIDER_GET_CUBES	BAPI_MDPROVIDER_GET_HIERARCHYS
BAPI_MDPROVIDER_GET_LEVELS	BAPI_MDPROVIDER_GET_MEMBERS
BAPI_MDPROVIDER_GET_VARIABLES	BAPI_MDDATASET_GET_CELL_DATA

BAPI_MDDATASET_DELETE_OBJECT	BAPI_MDDATASET_CREATE_OBJECT
BAPI_MDDATASET_SELECT_DATA	BAPI_MDPROVIDER_SET_KEY_DATE
BAPI_MDDATASET_GET_AXIS_DATA	BAPI_MDDATASET_GET_AXIS_INFO

SAP BW BEx Global Properties

This topic describes the SAP BW BEx global properties file that you can edit to control how TDV works with SAP BW BEx. Global properties apply to all instances of the SAP data source. You can edit the properties to control how TDV works with SAP BW BEx.

SAP BW BEx Global Trace Level Property File

Global properties are stored in the file:

<TDV_install_directory>\apps\dlm\app_ds_sapbw\conf\product.properties

Edit global properties using a text editor such as Notepad on Windows. TDV must be restarted for property changes to take effect.

SAP BW BEx has only one global property.

sap.jco.traceLevel

Default Value: 2

Description:

Turns on the SAP JCo trace. Allowed levels are 0 through 10. The commonly ones used are:

0—Nothing

1—Errors

2—Errors and warnings

3—Info messages, errors and warnings

4—Execution path, info messages, errors and warnings

5—Verbose execution path, info messages, errors and warnings

6—Verbose execution path, limited data dumps, info messages, errors and warnings

7—Full execution path, data dumps with metadata, verbose info messages, errors and warnings

8—Full execution path, full data dumps with metadata, verbose info messages, errors and warnings

Using Siebel with TDV

This topic describes how to use a Siebel data source.

You must have a Siebel account with API-level access.

Siebel has two-level hierarchy of resources:

- Business Objects—are represented in TDV as schemas.
- Business Components—are represented as database tables.

When you introspect Business Components, you automatically get these system fields: Created, Created By, Updated, and Updated By.

The Business Services folder, contains Business Services, represented as schemas, which contain Business Service Methods, represented as database procedures.

Siebel Limitations

Application Views for Siebel

The organization of the views and folders follows the hierarchy of the Siebel Repository: Business Objects and Business Components. Each view uses one or more Business Components from the Repository to create a view containing relevant business information. Application View field names are the same as the field names of the Business Components.

Filter Data from Application Views

You can use application views to filter Siebel data either with Siebel filters or with TDV filtering, depending on the view and the capabilities of Siebel. Siebel supports simple filters in a WHERE clause.

Siebel Basic Tab

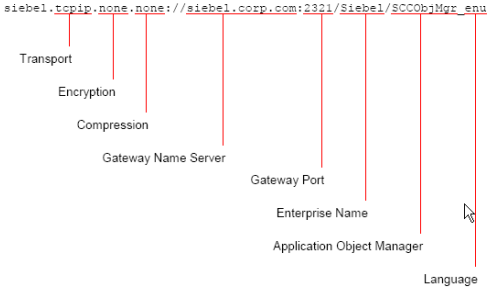
Input Field	Description
Server Name	Name of the computer on which the Siebel Server is installed.
Broker Port	The port that the Siebel Server listens on: 2321 by default for Siebel versions 7.7 and higher, 2320 by default for Siebel versions prior to 7.7.
Siebel Enterprise Name	The name of the Siebel Enterprise Server. The Siebel Enterprise Server is a name for a logical grouping of Siebel Servers that supports a group of Siebel users. The default name is siebel.
Application Object Manager	The name of the Siebel Application Object Manager (AOM). The AOMs available depend on the base application and the language. AOMs can be activated through the Siebel Server Manager. Example AOM: SCCObjMgr.
Language	The language used by the Siebel instance. The default is EN for English.
Username and Password	Valid user name and password for Siebel.
Save Password	This option works in combination with the Pass-through Login option. By default, this option is disabled and cannot be edited. It becomes editable when you select the Pass-through Login option. If you accept the default, the password is saved and the Pass-through Login option remains disabled. See the *TDV User Guide* for information on pass-through login. Use the query optimizer for statistics gathering.
Pass-through Login	Works in combination with the Save Password option. By default, this mode is Disabled (“non-pass-through mode”) and the password is saved. Refer to the details given for the Save Password option. If you select Enabled (“pass-through mode”), the Save Password option becomes editable. The operations you can and cannot perform in pass-through mode depend on whether Save Password is checked.

Save password?	Operations you can perform	Operations you cannot perform
Yes	Introspection. You do not have to resupply the password.	N/A
No	• Query/update/insert/delete operations. You need to resupply the original login credentials for the current session. • Reintrospection, Add/Remove data source resources. You will be prompted to resupply the same password that was used when the data source was originally introspected.	• Schedule reintrospection. • Statistics gathering, using the query optimizer.

Working with Siebel Connect Strings

A Siebel connect string contains most of the parameters necessary to communicate with Siebel. To use a Siebel connect string with TDV, you need to specify individual parameter values to put into the corresponding input fields of a Siebel data source. An appropriate connect string may already be available in a file named eapps.cfg on your Siebel Web server.

The connect string contains the parameters and their relationship to Siebel data source fields. The following connect string applies to Siebel versions 7.7 and later. Earlier Siebel versions include the Siebel server name at the end.



SQL Support Reference for Siebel

This topic describes SQL support for Siebel, and for Siebel Multi-Valued Groups, which manage data relationships.

Understanding Siebel Resources

TDV integrates with Siebel at the level of data objects and services. In Siebel, objects and services are called Business Components and Business Services, respectively. The Siebel Adapter enables Siebel Business Components and Business Services to be used as fully SQL-92-compliant relational data sources.

Topic	Description of what the topic covers
Siebel Resources within TDV, page 152	describes the list of Business Components and Business Services, allows the user to select them and translates the Siebel metadata into relational tables that TDV understands.
Supported Capabilities, page 154	describes how a SQL statement is divided among Siebel TDV and other data sources that might be referenced in the query.
Joins and Query Performance, page 155	describes how joins can affect query performance, and discusses options for improving performance.
Timestamp Operations, page 155	describes how some SQL queries using time stamp operations might need to be modified for use with TDV.

Siebel Resources within TDV

The following sections describe how Siebel resources work within TDV:

Resource Hierarchy

The Siebel Adapter provides Siebel Business Components and Business Services as resources in TDV. Business Components are introspected by starting with the folder named "Business Objects" and drilling down into a two-level hierarchy of folders. The first level is the Business Object, which is a grouping of Business Components. If you are not sure which Business Object contains the Business Component you need, you can use Siebel Tools to find it.

Introspection automatically includes these system fields for Business Components: Created, Created By, Updated, and Updated By.

Business Services are introspected starting with the folder named "Business Services" and drilling down into a two-level hierarchy of folders. The first level is Business Service, which contains a collection of Business Service Methods.

Metadata Mapping

When introspecting Business Components, each field of the Business Component becomes a column in TDV with the same name. When introspecting Business Services, each parameter of the Business Service Method becomes a parameter in the TDV procedure with the same name. In both cases, data types in Siebel are mapped to TDV SQL-based types. The following table lists the Siebel data types, whether they are supported in TDV, and their corresponding TDV type.

Siebel Data Type Name	Supported?	TDV Data Type
TEXT, ID, PHONE, NOTE, BOOL	Yes	VARCHAR. If length is undefined in Siebel, it defaults to 255.
INTEGER	Yes	BIGINT
NUMBER, CURRENCY	Yes	DOUBLE
DATE	Yes	DATE
TIME	Yes	TIME
UTCDATETIME	Yes	TIMESTAMP
Link Specification	Yes	Inherits type of linked business component.
Multi-valued Field	Yes	Inherits type of associated business component.
String	Yes	VARCHAR(65536)
Integration Object	Yes	XML

Some field definitions, particularly link specifications, contain no type information in Siebel. TDV considers untyped fields to be VARCHAR(255). The data types of multi-valued and link specification fields are obtained from their associated Business Components, and MULTI_VALUED or LINK_SPECIFICATION is appended to native type names.

Supported Capabilities

Capabilities characterize the features and limitations of data sources. For example, an Oracle data source can execute subqueries, while Siebel cannot. Capabilities are consulted when a query is processed so that data sources receive only the query processing work they support; otherwise, TDV performs the work itself.

The following table lists commonly used capabilities and whether they are supported in queries against Siebel Business Components. Pushed indicates whether query processing can be passed to Siebel. For efficient queries, minimize use of non-push capabilities.

Capability	Supported?	Pushed?	Notes
CASE	Yes	No	
DELETE	Yes	Yes	Special behavior for multi-valued groups.
DISTINCT	Yes	No	
Filter	Yes	Yes	
Filter - LIKE	Yes	Yes	
Filter – BETWEEN	Yes	Yes	
Filter – IN	Yes	Yes	
Functions – Aggregate	Yes	No	
Functions – CAST	Yes	Yes	
Functions – Others	Yes	No	
GROUP BY	Yes	No	
INSERT	Yes	Yes	Special behavior for multi-valued groups.
Join	Yes	No	

Capability	Supported?	Pushed?	Notes
ORDER BY	Yes	Yes	
Subquery	Yes	No	
Transactions	No	No	
UNION	Yes	No	
UPDATE	Yes	Yes	Special behavior for multi-valued groups.

Joins and Query Performance

Joins cannot be pushed to Siebel. Executing joins in TDV can degrade performance, because a table scan would be required to fetch every row of the joined tables. The technology that TDV uses to connect with Siebel is not optimized for large data sets, so table scans should be avoided. It is better to use filters on queries to Siebel Business Components.

A semijoin is the best way to reduce the number of Siebel rows retrieved and processed. To force a semijoin to occur in a query, add the option immediately before the table to be joined. For example:

```
SELECT * FROM A INNER {OPTION SEMIJOIN} JOIN B ON A.K = B.K
```

Values of A.K are collected and passed in a query to B as the filter:

```
SELECT * FROM A
SELECT * FROM B WHERE K IN ({values_of_X.K_from_previous_query})
```

If A has many rows, queries against B can be lengthy. TDV automatically partitions them and reassembles the results into a unified set.

Put the table that returns the larger number of rows on the right side of the join whenever possible. When running a new query for the first time, activate the Execution Plan tab in Studio and click Execute and Show Statistics. Examine each node's row count and query after processing has begun to make sure filters are pushed down to Siebel. This is a good way to see the mechanics of a semijoin in action. If the interaction between TDV and Siebel is still unclear and performance is poor, enable debug logging for the adapter as described in the *TDV Installation and Upgrade Guide*.

Timestamp Operations

Some Siebel time stamp operations might need to be modified due to TDV's more stringent time stamp requirements. For example, you might have this Siebel time stamp operation:

```
WHERE Service_Request."Opened Date" = date '2015-11-02'
```

You must change this to:

```
WHERE Service_Request."Opened Date" BETWEEN date '2015-11-02' AND
date '2015-11-03'
```

Siebel Multi-Valued Groups

To manage one-to-many or many-to-many relationships, Siebel Business Components use Multi-Valued Groups (MVG). Multiple records may match a given field or set of fields in a record. An example is a Contact with multiple Addresses. Siebel uses MVGs to manage the link between the Contact and Addresses. Operations on Siebel MVGs are inferred automatically from SQL statements and Siebel metadata. Specific behavior is based on the type of SQL operation executed.

Filter on MVG Fields

TDV uses a Siebel EXISTS clause to expand queries with WHERE clauses that reference an MVG field. For example, take the following TDV SQL query:

```
SELECT * FROM /Siebel/Contact/Contact
WHERE "Related Contact UId" = '1-16N'
```

This would be provided to Siebel as the following search expression:

```
EXISTS("Related Contact UId" = '1-16N')
```

SELECT from 1:n (one-to-many) MVG Fields

Business Components with 1:n (one-to-many) relationships to other Business Components can typically be queried by combining simple SQL with filters on foreign keys. For example, you can find all of the contacts associated with a given account as follows:

```
SELECT * FROM /Siebel/Contact/Contact WHERE "Account Id" = '1-16N'
```

SELECT from m:n (many-to-many) MVG Fields

Business Components with m:n (many-to-many) relationships cannot be completely queried in SQL. Siebel maintains an intersection table that maps the MVGs, but it is not accessible directly to Business Components. (Queries are limited to fields available in any given Business Component.) For example, contacts can have multiple addresses, and addresses can be associated with multiple contacts. The Contact Business Component designates a single address as primary. A query for a contact and address would return the contact and address identified as the primary address.

INSERT with VALUES for MVG Fields

MVG fields are automatically associated with records matching the values provided in the INSERT statement. For example, consider the following SQL:

```
INSERT INTO /Siebel/Contact/Contact ("First Name", "Last Name",
"Personal Street Address", "Personal City") VALUES ('Tom',
'Siebel', '1 Siebel Way', 'San Mateo');
```

This creates a new record in the Contact Business Component, and examines all MVG relationships in the fields provided (in this case, Personal Street address and Personal City). If the MVG link is specified as “No Associate” in Siebel’s metadata, the fields are directly set on the parent (Contact) Business Component. Otherwise, a search is performed on the child Business Component (Address) for records matching the inserted values (1 Siebel Way, San Mateo). The first matching address record is associated with the new contact record. If no matching address records are found, a new address record is added and associated

Note: In the Siebel UI, an additional step is sometimes performed whereby an Address or other MVG receives a Primary designation This step is not performed by the Siebel data source. To set a related record as Primary, that record’s unique identifier must be queried and its corresponding Primary field set manually with an UPDATE statement on its parent.

UPDATE with SET on MVG Fields

UPDATE on MVG fields behaves the same as INSERT. An UPDATE cannot safely modify a record in a linked Business Component because other records may depend on it. Instead, perform a search on the linked Business Component to find a record matching the values specified in the SET fields. Add the first record found to the association list for the record. If no matching record is found, add a new record to the association list. Existing associations remain unchanged.

DELETE on MVG Fields

A DELETE operation on a Business Component never cascades to remove associated Business Components. For example, if an account is inserted with a new address, that address is added to the proper MVG business component and a relationship established between the account and address. If this account is later deleted, the address record remains in Siebel.

TDV Apache HBase Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to HBase

The HBase Adapter connects to Apache HBase via the HBase REST (Stargate) server. Set the Port and Server properties to connect to HBase. The Server property will typically be the host name or IP address of the server hosting Apache HBase. If there are multiple nodes, you will use the host name or IP address of the machine running the REST (Stargate) server.

Starting the Server

Different Hadoop distributions contain different interfaces and means of starting and stopping the HBase REST server, along with different default port settings. In most distributions, the HBase REST server can be started in the foreground by running the following command: "hbase rest start -p <port>". Please consult your Hadoop distribution's documentation for further information regarding the HBase REST server.

Authenticating to HBase

If authentication is configured for your server, set AuthScheme to NEGOTIATE and set the User and Password, if necessary, to authenticate through Kerberos.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Scheme	The scheme used for authentication. Accepted entries are NONE, BASIC, and NEGOTIATE (Kerberos). NONE is the default.
Datetime Format	The format used when inserting datetime value into the database.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Kerberos KDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
Kerberos Realm	The Kerberos Realm used to authenticate the user with.
Kerberos SPN	The Service Principal Name for the Kerberos Domain Controller.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per page from Apache HBase.
Password	The password used to authenticate to Apache HBase.
Port	The port for the Apache HBase REST server.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.

Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Apache HBase from the provider.
Row Scan Depth	The number of rows to scan to determine columns for the table.
Server	The host name or IP address of the Apache HBase REST server.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Type Detection Scheme	Determines how to determine the data type of columns.
User	The user who is authenticating to Apache HBase.

Auth Scheme

The scheme used for authentication. Accepted entries are NONE, BASIC, and NEGOTIATE (Kerberos). NONE is the default.

Data Type

string

Default Value

"NONE"

Remarks

This field is used to authenticate against the server. Use the following options to select your authentication scheme:

NONE: No authentication is performed.

BASIC: Basic authentication is performed.

NEGOTIATE: If AuthScheme is set to NEGOTIATE, the adapter will negotiate an authentication mechanism with the server. Set AuthScheme to NEGOTIATE if you want to use Kerberos authentication.

Datetime Format

The format used when inserting datetime value into the database.

Data Type

string

Default Value

"yyyy-MM-dd'T'HH:mm:ss.fffzzz"

Remarks

This can be used to automatically format any datetime values entering a database.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to HBase and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Kerberos KDC

The Kerberos Key Distribution Center (KDC) service used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos Key Distribution Center (KDC) service. The Kerberos Key Distribution Center (KDC) service is conventionally colocated with the domain controller. If Kerberos KDC is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using the config file krb5.conf, or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the configured domain name and host as a last resort.

Note: Windows authentication is supported in JRE 1.6 and above only.

Kerberos Realm

The Kerberos Realm used to authenticate the user with.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using SPNEGO or Windows Authentication. The Kerberos Realm is used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name. If Kerberos Realm is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using a config file (krb5.conf) or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the user-configured domain name and host as a last resort. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

Kerberos SPN

The Service Principal Name for the Kerberos Domain Controller.

Data Type

string

Default Value

""

Remarks

If the Service Principal Name on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, set the Service Principal Name here.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per page from Apache HBase.

Data Type

string

Default Value

"10000"

Remarks

The PageSize can control the number of results returned per page from Apache HBase. Setting a higher PageSize will cause more data to come back in a single HTTP request, but may take longer to execute. Setting a smaller PageSize will increase the number of HTTP requests to get all the data, but is generally recommended to ensure timeout exceptions do not occur.

Password

The password used to authenticate to Apache HBase.

Data Type

string

Default Value

""

Remarks

The User, Password, and AuthScheme are together used to authenticate with the server.

The password used to authenticate to Apache HBase.

Port

The port for the Apache HBase REST server.

Data Type

string

Default Value

"8080"

Remarks

The port for the Apache HBase REST (Stargate) server.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.

TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Apache HBase from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Row Scan Depth

The number of rows to scan to determine columns for the table.

Data Type

string

Default Value

"100"

Remarks

The number of rows to scan to determine columns for the table. Higher values will result in a longer request, but will be more accurate.

Server

The host name or IP address of the Apache HBase REST server.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the Apache HBase REST (Stargate) server.

To use SSL, you can prefix the host name or IP address with 'https://'.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZI hv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70 e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f8 01cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Type Detection Scheme

Determines how to determine the data type of columns.

Data Type

string

Default Value

"RowScan"

Remarks

None	Setting TypeDetectionScheme to None will return all columns as strings. The RowScanDepth determines the number of rows to be scanned to retrieve the column listing.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned to retrieve the column listing and column data type.

User

The user who is authenticating to Apache HBase.

Data Type

string

Default Value

""

Remarks

The user who is authenticating to Apache HBase.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the HBase Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Fine-Tuning Data Access

You can use the following properties to configure automatic data type detection, which is enabled by default.

TypeDetectionScheme: You can use this property to enable or disable automatic type detection based on the value specified in RowScanDepth.

RowScanDepth: This property determines the number of rows that will be scanned to determine column data types.

The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

HTTP Proxies

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The HBase Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the HBase API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the HBase adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | AND | OR | LIKE
| NOT LIKE | IN | NOT IN } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Account
```

Rename a column:

```
SELECT "Name" AS MY_Name FROM Account
```

Search data:

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

The HBase APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, AND, OR, LIKE, NOT LIKE, IN, NOT IN.

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Name INTO "csv://Account.txt" FROM "Account" WHERE
Industry = 'Floppy Disks'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Name INTO "csv://Account.txt;delimiter=tab" FROM
"Account" WHERE Industry = 'Floppy Disks'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "Floppy Disks");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Account SET Name='Floppy Disks' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:apachehbase:Server=127.0.0.1;Por
t=8080;",);
String cmd = "DELETE FROM Account WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The HBase Adapter models the HBase REST APIs as relational tables and stored procedures that can be accessed with standard SQL. This enables access from standards-based tools.

The table definitions are dynamically retrieved. When you connect, the adapter connects to HBase and gets the list of tables and the metadata for the tables by querying the HBase REST server. Any changes to the remote data are immediately reflected in your queries.

Stored Procedures are function-like interfaces to the data source. Stored procedures model actions that typically cannot be represented as SELECT, INSERT, UPDATE, or DELETE statements. The stored procedures of the adapter surface the capabilities of the DDL (data definition language) in HBase.

A full list is available upon customer request.

TDV Active Directory Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to Active Directory

To authenticate requests, set the User and Password properties to valid Active Directory credentials (e.g., set User to "Domain\\BobF" or "cn=Bob F,ou=Employees,dc=Domain").

The adapter uses plaintext authentication by default, since the adapter attempts to negotiate TLS/SSL with the server. You can specify another authentication method with AuthMechanism.

See Advanced Settings for more information on TLS/SSL configuration.

Connecting to Active Directory

Set Server and Port for basic connectivity. Additionally, you can fine-tune the connection with the following:

FollowReferrals: When set, the adapter surfaces data as views from only referral servers. To modify data on a referral server, you must specify this server with Server and Port.

LDAPVersion: Set this to the version of the protocol your server implements; by default, the adapter uses version 2.

UseDefaultDC: Set this to connect to the default Domain Controller and authenticate using the current user credentials.

Fine-Tuning Data Access

The following properties control the scope of data returned:

BaseDN will limit the scope of LDAP searches to the height of the distinguished name provided. *Note*: Specifying a narrow BaseDN may greatly increase performance; for example, a value of "cn=users,dc=domain" will only return results contained within "cn=users" and its children.

Scope: This property enables more granular control over the data to return from a subtree.

Customizing Tables

The adapter surfaces the columns most often needed from Active Directory entities. However, if you need to work with other data, the tables are easy to modify. Tables are defined in schema files, which have a simple format.

See Working with Active Directory Tables for a guide to extending the default schemas or writing your own. To use custom schemas, set the Location property to the folder containing the schema files.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Mechanism	The authentication mechanism to be used when connecting to the Active Directory server.
Base DN	The base portion of the distinguished name, used for limiting results to specific subtrees.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Follow Referrals	Whether or not to follow referrals returned by the Active Directory server.

LDAP Version	The LDAP version used to connect to and communicate with the server.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password for the distinguished name of the specified user.
Port	The port the Active Directory server is running on.
Readonly	You can use this property to enforce read-only access to ActiveDirectory from the provider.
Scope	Whether to limit the scope of the search to the whole subtree (BaseDN and all of its descendants), a single level (BaseDN and its direct descendants), or the base object (BaseDN only).
Server	The domain name or IP of the Active Directory server.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Default DC	Used to connect to the default Domain Controller and authenticate using the current user credentials.
User	The distinguished name of a user.

Auth Mechanism

The authentication mechanism to be used when connecting to the Active Directory server.

Data Type

string

Default Value

"SIMPLE"

Remarks

By default, AuthMechanism is SIMPLE, and default plaintext authentication is used to log in to the server. If AuthMechanism is set to DIGESTMD5, the more secure DIGEST-MD5 authentication is used. If AuthMechanism is set to NEGOTIATE, NTLM/Negotiate authentication will be used.

SIMPLE

DIGESTMD5

NEGOTIATE

Base DN

The base portion of the distinguished name, used for limiting results to specific subtrees.

Data Type

string

Default Value

""

Remarks

Specifying a base DN may greatly improve performance when returning entries for large servers by limiting the number of entries that need to be examined.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Active Directory and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Follow Referrals

Whether or not to follow referrals returned by the Active Directory server.

Data Type

bool

Default Value

false

Remarks

When following referrals, you will only be able to return data from the referral servers. INSERT/UPDATE/DELETE will not be available without updating the connection string to connect directly to that server.

LDAP Version

The LDAP version used to connect to and communicate with the server.

Data Type

string

Default Value

"2"

Remarks

Valid options are 2 and 3 for LDAP versions 2 and 3.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password for the distinguished name of the specified user.

Data Type

string

Default Value

""

Remarks

Together with User, this field is used to authenticate against the Active Directory server.

Port

The port the Active Directory server is running on.

Data Type

string

Default Value

"389"

Remarks

The port the Active Directory server is running on. Together with Server, this property is used to specify the Active Directory server.

Readonly

You can use this property to enforce read-only access to ActiveDirectory from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Scope

Whether to limit the scope of the search to the whole subtree (BaseDN and all of its descendants), a single level (BaseDN and its direct descendants), or the base object (BaseDN only).

Data Type

string

Default Value

"WHOLESUBTREE"

Remarks

Whether to limit the scope of the search to the whole subtree (BaseDN and all of its descendants), a single level (BaseDN and its direct descendants), or the base object (BaseDN only). Limiting scope can greatly improve the search performance.

WholeSubtree

SingleLevel

BaseObject

Server

The domain name or IP of the Active Directory server.

Data Type

string

Default Value

""

Remarks

Note: This does not need to include the LDAP:\\ portion, only the server domain name or IP.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZI hv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70 e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f8 01cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Default DC

Used to connect to the default Domain Controller and authenticate using the current user credentials.

Data Type

bool

Default Value

false

Remarks

Used to connect to the default Domain Controller and authenticate using the current user credentials.

User

The distinguished name of a user.

Data Type

string

Default Value

""

Remarks

Together with Password, this field is used to authenticate against the Active Directory server.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Active Directory Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Working with Active Directory Tables

The adapter includes table schemas for many standard Active Directory objects. You can easily extend the included table schemas to edit column behavior or you can write your own from scratch.

Table schemas are defined in .rsd files, which are simple configuration files. This section will walk through different parts of the schema, adding several columns to the Person table as an example.

You can find the Person.rsd file in the db subfolder in the installation folder of the Active Directory Adapter.

Connecting to Custom Tables

To use custom schemas, set the Location property to the folder containing the schema files.

Defining a New Table

It is important to define a new table with the same name as the object class that the table will represent. This will allow the adapter to search for only the desired object class when querying the Active Directory server. The file name defines the table name.

Defining Table Columns and Inputs

Columns are defined in the *rsb:info* block, a shown below. The *attr* tags in the schema represent the columns of the table. These should match the attributes that make up the desired object class.

There are a few columns that every table should include, regardless of the object class:

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">
<rsb:info title="Person" description="Create, update, delete, and
query person entries in Active Directory.">
<!-- Required Columns -->
<attr name="Id"          xs:type="string" readonly="true"
key="true"        />
<attr name="DN"          xs:type="string" readonly="true"
required="false"  />
<attr name="RDN"         xs:type="string" readonly="true"
required="false" />
<attr name="BaseDN"      xs:type="string" readonly="true"
required="false"  />
```

Note: The *title* attribute of the *rsb:info* block must match the name of the .rsd file.

Customizing Column Behavior

Each column requires at least *name* and *xs:type* attributes. Additionally, you will need to specify *dataFormat* to decide how data is returned from the table. For example:

```
<!-- Person Required Attributes -->
<attr name="ObjectClass"
other:dataFormat="splitDataByRow" xs:type="string"
readonly="false" required="false" />
<attr name="SN"
other:dataFormat="delimitedData"  xs:type="string"
readonly="false" required="false" />
<attr name="CN"
other:dataFormat="delimitedData"  xs:type="string"
readonly="false" required="false" />

<!-- Person Optional Attributes -->
<attr name="UserPassword"
other:dataFormat="delimitedData"  xs:type="string"
readonly="false" required="false" />
```

```
<attr name="TelephoneNumber"
other:dataFormat="delimitedData"  xs:type="string"
readonly="false" required="false" />
<attr name="SeeAlso"
other:dataFormat="delimitedData"  xs:type="string"
readonly="false" required="false" />
<attr name="Description_1"
other:dataFormat="splitDataByCol" xs:type="string"
readonly="false" required="false" />
<attr name="Description_2"
other:dataFormat="splitDataByCol" xs:type="string"
readonly="false" required="false" />
<attr name="Description_3"
other:dataFormat="splitDataByCol" xs:type="string"
readonly="false" required="false" />
```

The *other:dataFormat* attribute has three options:

delimitedData: Return multiple Active Directory attribute values as delimited strings, separated by the *delimiter* character defined in the Table Settings section of the .rsd file, detailed later.

This is the default format in which to retrieve data and the delimiter defaults to a semicolon.

pushDataByRow: Push multiple Active Directory attribute values for the same DN as separate rows. All other columns will be pushed consistently, and the index in Id will be incremented. *Note*: Pushing multiple columns like this will exponentially grow the result set, potentially causing performance issues.

pushDataByCol: Push multiple Active Directory attribute values for the same DN with an appended index on the column name. You need to define multiple columns and append an "_n" to the end; for example, ObjectClass_1, ObjectClass_2, and ObjectClass_3. In this example, if there are more than 3 values, the remaining values will not be visible in the table, unless more columns are added.

Example: Splitting the ObjectClass Attribute

The code below can be used to split the different values of the *ObjectClass* attributes into their own rows and *Description* attributes into their own columns. Notice the column definition now includes multiple columns for the *Description* attribute. Also note the *other:dataFormat* attribute for the *attr*.

```
...
<attr name="ObjectClass"     other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="SN"              other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="CN"              other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="UserPassword"    other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
```

```
<attr name="TelephoneNumber" other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="SeeAlso"          other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="Description_1"   other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="Description_2"   other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />
<attr name="Description_3"   other:dataFormat="delimitedData"
xs:type="string"  readonly="false" required="false" />

</rsb:info>

<!-- Table Settings -->
<rsb:set attr="delimiter" value=";"/>
...
```

An example result will look like:

Id	**DN**	**Object Class**	**SN**	**CN**	UserPassword	TelephoneNumber	SeeAlso	Description_1	Description_2	Description_3
1|CN=User1,DC=Test	CN=User1,DC=Test	Top	TestSN	User1		555-5555	A;B;C	Desc1	Desc2	Desc3
2|CN=User1,DC=Test	CN=User1,DC=Test	User	TestSN	User1		555-5555	A;B;C	Desc1	Desc2	Desc3

Specifying Column Encoding

In addition to data format on inputs, encoding can also be specified. Currently, returning data with UTF8 encoding or BASE64 encoding is supported. In order to retrieve data with a specified encoding, the *other:encoding* field must be specified for the desired attribute to be encoded. If no encoding is specified, UTF8 is the default.

An example of specifying encoding for an attribute:

```
...
```

```
<attr name="ObjectClass"     other:dataFormat="delimitedData"
other:encoding="UTF8"   xs:type="string"  readonly="false"
required="false" desc="The object class of the entry."/>
<attr name="SN"              other:dataFormat="delimitedData"
other:encoding="BASE64" xs:type="string"  readonly="false"
required="false" desc="The surname of the person."/>
...
```

Configuring Table Settings

In addition to the attributes and inputs, you will need to specify the *delimiter*.

The *delimiter* specifies the character that will be used for delimited data. Delimited data will be returned for any attribute that appears multiple times for a single object (unless otherwise specified in *other:dataFormat*).

For example, the code below will concatenate multiple values of an attribute using the ';' character.

```
...
</rsb:info>

<!-- Table Settings -->
<rsb:set attr="delimiter" value=";"/>
...
```

Defining Supported Operations

Operation definitions will remain exactly the same for all newly created tables: Simply copy and paste these from an existing table, as needed.

```
<!-- Operation definitions -->
<rsb:script method="GET">
<rsb:set attr="action" value="Get" />
<rsb:call op="adadoAD" out="toout" ignoreprefix="ldap">
<rsb:push item="toout"/>
</rsb:call>
</rsb:script>

<rsb:script method="POST">
<rsb:set attr="action" value="Post" />
<rsb:call op="adadoAD" out="toout" ignoreprefix="ldap">
<rsb:push item="toout"/>
</rsb:call>
</rsb:script>

<rsb:script method="MERGE">
<rsb:set attr="action" value="Merge" />
<rsb:call op="adadoAD" out="toout" ignoreprefix="ldap">
<rsb:push item="toout"/>
</rsb:call>
</rsb:script>

<rsb:script method="DELETE">
```

```
<rsb:set attr="action" value="Delete" />
<rsb:call op="adadoAD" out="toout" ignoreprefix="ldap">
<rsb:push item="toout"/>
</rsb:call>
</rsb:script>
```

Advanced Settings

The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The Active Directory Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Active Directory API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Active Directory adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
```

```
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | >= | <= | != | LIKE | AND | OR } [
<expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM User
```

Rename a column:

```
SELECT "CN" AS MY_CN FROM User
```

Search data:

```
SELECT * FROM User WHERE CN = 'Administrator';
```

The Active Directory APIs support the following operators in the WHERE clause: =, >=, <=, !=, LIKE, AND, OR.

```
SELECT * FROM User WHERE CN = 'Administrator';
```

Sort a result set in ascending order:

```
SELECT Id, CN FROM User  ORDER BY CN ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, CN INTO "csv://User.txt" FROM "User" WHERE CN =
'Administrator'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, CN INTO "csv://User.txt;delimiter=tab" FROM "User" WHERE
CN = 'Administrator'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO User (CN) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
```

```
pstmt.setString(1, "User Name");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE User SET CN='User Name' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "CN=User Name,CN=Users,DC=Domain");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]
```

```
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:activedirectory:user=MyUserName;
password=MyPassword;Server=MyServer;Port=MyPort;",);
String cmd = "DELETE FROM User WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "CN=User Name,CN=Users,DC=Domain");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The Active Directory Adapter models Active Directory entities in relational tables and stored procedures.

A full list is available upon customer request.

TDV Couchbase Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to Couchbase

To connect using the Login method, set User, Password, and Server to the credentials for the account and the address of the server you want to connect to.

Optionally, set N1QLPort if you are not connecting to the default port, 8093.

Securing Couchbase Connections

You can set UseSSL to negotiate an SSL exchange with the server. This property overrides the default N1SQL port the adapter connects to; when UseSSL is set, the adapter connects on port 18093 by default.

By default, the adapter uses the trusted certificate store configured for the system. You can specify another certificate store with SSLServerCert.

Accessing NoSQL Tables

The adapter implements Automatic Schema Discovery that is highly configurable. The following sections outline the adapter's defaults and link to ways to further customize.

Flattening Nested JSON

By default, the adapter projects columns over the properties of objects. Arrays are returned as JSON strings, by default. You can use the following properties to access array elements, including objects nested in arrays.

FlattenArrays: Set this property to the number of array elements that you want to return as column values. You can also use this property with FlattenObjects to extract the properties of objects nested in arrays.

FlattenObjects: By default, this is true; that is, the properties of objects and nested objects are returned as columns. When you set FlattenArrays, objects nested in the specified array elements are also flattened and returned as columns.

Other mechanisms for accessing nested objects are detailed in NoSQL Database.

Advanced Tab

Connection String Options

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Credentials File	A file containing a list of credentials to access Couchbase. This takes priority over other forms of Authentication.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Flatten Arrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
Flatten Objects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Generate Schema Files	Indicates the user preference as to when schemas should be generated and saved.
Infer Num Sample Values	The maximum number of values for every field to scan before determining its data type.
Infer Sample Size	The maximum number of documents to scan for the columns available in the bucket.
Infer Similarity Metric	Specifies the similarity degree where different schemas will be considered to be the same flavor.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
N1QL Port	The port for connecting to the Couchbase N1QL Endpoint. Default 8093 for plaintext and 18093 for SSL.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per page of data retrieved from the Couchbase server.
Password	The password used to authenticate to Couchbase.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.

Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Couchbase from the provider.
Row Scan Depth	The maximum number of documents to scan for the columns available in the bucket.
Schema Depth	An integer value specifying the depth at which objects fields will stop being scanned for more columns and instead will be turned to aggregate.
Select Array Indexes	This option enables or disables selecting individual array indexes when selecting a column seperated by the PeriodsSeparator.
Server	The address of the Couchbase server to which you are connecting.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Type Detection Scheme	Comma separated list of options for how the provider will scan the data to determine the fields and datatypes for the bucket.
Use Infer	Specifies whether to detect table columns using Couchbases Infer command. This option is not supported in Couchbase community servers and versions lower than 4.5.1 will be defaulted back to the value set in TypeDetectionScheme .
User	The user who is authenticating to Couchbase.
Use SSL	This property is used to determine the port for connecting to the Couchbase N1QL Endpoint. If false, the default, default ports will be assumed, 8093 for plaintext and 18093 for SSL, based on the value specified in UseSSL.
Web Console Port	The port for connecting to the Couchbase Web Console. Default 8091 for plaintext and 18091 for SSL.

Credentials File

A file containing a list of credentials to access Couchbase. This takes priority over other forms of Authentication.

Data Type

string

Default Value

""

Remarks

A file containing a list of credentials to access Couchbase. This takes priority over other forms of Authentication.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Couchbase and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Flatten Arrays

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.

Data Type

string

Default Value

"0"

Remarks

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

Set FlattenArrays to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

For example, you can return an arbitrary number of elements from an array of strings:

```
["FLOW-MATIC","LISP","COBOL"]
```

When FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value

languages_0	FLOW-MATIC

Flatten Objects

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Data Type

bool

Default Value

true

Remarks

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON. The property name is concatenated onto the object name with an underscore to generate the column name.

For example, you can flatten the nested objects below at connection time:

```
address : {
  "street" : "123 Main St.",
  "city"   : "Nowhere",
  "state"  : "NY",
  "zip"    : "12345"
}
```

When FlattenObjects is set to true, the preceding object is flattened into the following table:

Column Name	Column Value
address_street	123 Main St.
address_city	Nowhere
address_state	NY
address_zip	12345

Generate Schema Files

Indicates the user preference as to when schemas should be generated and saved.

Data Type

string

Default Value

"Never"

Remarks

GenerateSchemaFiles enables you to save the table definitions identified by Automatic Schema Discovery. This property outputs schemas to .rsd files in the path specified by Location.

Available settings are the following:

Never: A schema file will never be generated.

OnUse: A schema file will be generated the first time a table is referenced, provided the schema file for the table does not already exist.

OnStart: A schema file will be generated at connection time for any tables that do not currently have a schema file.

Note that if you want to regenerate a file, you will first need to delete it.

Generate Schemas with SQL

When you set GenerateSchemaFiles to **OnUse**, the adapter generates schemas as you execute SELECT queries. Schemas are generated for each table referenced in the query.

Generate Schemas on Connection

Another way to use this property is to obtain schemas for every table in your database when you connect. To do so, set GenerateSchemaFiles to **OnStart** and connect.

Editing Schemas

Schema files have a simple format that makes them easy to modify. See Custom Schema Definitions for more information.

Infer Num Sample Values

The maximum number of values for every field to scan before determining its data type.

Data Type

string

Default Value

"10"

Remarks

The maximum number of values for every field to scan before determining its data type.

Infer Sample Size

The maximum number of documents to scan for the columns available in the bucket.

Data Type

string

Default Value

"100"

Remarks

Since Couchbase is schemaless, this property controls the maximum number of values of each field that will be scanned to determine datatype of a column.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

Infer Similarity Metric

Specifies the similarity degree where different schemas will be considered to be the same flavor.

Data Type

string

Default Value

"0.7"

Remarks

Specifies the similarity degree where different schemas will be considered to be the same flavor.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

N1QL Port

The port for connecting to the Couchbase N1QL Endpoint. Default 8093 for plaintext and 18093 for SSL.

Data Type

string

Default Value

""

Remarks

The port for connecting to the Couchbase N1QL Endpoint. If empty, default ports will be assumed, 8093 for plaintext and 18093 for SSL, based on the value specified in UseSSL.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per page of data retrieved from the Couchbaseserver.

Data Type

string

Default Value

"1000"

Remarks

The number of results to return per page of data retrieved from the Couchbase server.

Password

The password used to authenticate to Couchbase.

Data Type

string

Default Value

""

Remarks

The password used to authenticate to Couchbase.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.

TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Couchbase from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Row Scan Depth

The maximum number of documents to scan for the columns available in the bucket.

Data Type

string

Default Value

"100"

Remarks

Since Couchbase is schemaless, this property controls the maximum number of documents that will be scanned to determine columns for the bucket.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

Schema Depth

An integer value specifying the depth at which objects fields will stop being scanned for more columns and instead will be turned to aggregate.

Data Type

int

Default Value

99

Remarks

This option defines the columns as *parent*.Column, this carries on as long as the depth allows every column carrying on the previous parent.

Select Array Indexes

This option enables or disables selecting individual array indexes when selecting a column seperated by the PeriodsSeparator.

Data Type

bool

Default Value

false

Remarks

You can use this as: SELECT `Users.0.Name` FROM Departments. This will allow you to select a specific user inside an object array.

Server

The address of the Couchbase server to which you are connecting.

Data Type

string

Default Value

""

Remarks

The address of the Couchbase server to which you are connecting.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Type Detection Scheme

Comma separated list of options for how the provider will scan the data to determine the fields and datatypes for the bucket.

Data Type

string

Default Value

"RowScan"

Remarks

None	Setting TypeDetectionScheme to None will return all columns as string type. Cannot be used with other options.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned. Can be used with Recent.
Recent	Setting TypeDetectionScheme to Recent will determine whether RowScan is executed on the most recent documents in the collection. Can be used with RowScan.

Use Infer

Specifies whether to detect table columns using Couchbases Infer command. This option is not supported in Couchbase community servers and versions lower than 4.5.1 will be defaulted back to the value set in TypeDetectionScheme .

Data Type

bool

Default Value

true

Remarks

Used in conjunction with SchemaDepth, InferNumSampleValues, InferSimilarityMetric and InferSampleSize determines a schema for a bucket.

User

The user who is authenticating to Couchbase.

Data Type

string

Default Value

""

Remarks

The user who is authenticating to Couchbase.

Use SSL

This property is used to determine the port for connecting to the Couchbase N1QL Endpoint. If false, the default, default ports will be assumed, 8093 for plaintext and 18093 for SSL, based on the value specified in UseSSL.

Data Type

bool

Default Value

false

Remarks

This property is used to determine the port for connecting to the Couchbase N1QL Endpoint. If false, the default, default ports will be assumed, 8093 for plaintext and 18093 for SSL, based on the value specified in UseSSL.

Web Console Port

The port for connecting to the Couchbase Web Console. Default 8091 for plaintext and 18091 for SSL.

Data Type

string

Default Value

""

Remarks

The port for connecting to the Couchbase Web Console. If left empty, default ports will be assumed, 8091 for plaintext and 18091, based on the value specified in UseSSL.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Couchbase Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Fine Tuning Data Access

You can use the following properties to gain greater control over Couchbase API features and the strategies the adapter uses to surface them:

TypeDetectionScheme: You can use this property to enable or disable automatic data type detection based on the value specified in RowScanDepth.

RowScanDepth: This property determines the number of rows that will be scanned to detect column data types when generating table metadata.

QueryPassthrough: This property enables you to execute N1QL queries through the adapter. See Query Mapping

for N1QL examples and the corresponding SQL.

UseInfer: You can use this property to enable or disable the Infer command in order to expose different Couchbase bucket flavors as individual tables.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the connection properties needed to authenticate and connect. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

NoSQL Database

Couchbase is a schema-free document database that provides high performance, availability, and scalability. These features are not necessarily incompatible with a standards-compliant query language like SQL-92. In this section we will show various schemes that the adapter offers to bridge the gap with relational SQL and a document database.

The adapter models the schema-free Couchbase objects into relational tables and translates SQL queries into N1QL queries to get the requested data. The adapter offers two ways, Automatic Schema Discovery and Custom Schema Definitions, to model Couchbase documents as tables.

The Automatic Schema Discovery scheme automatically finds the data types in a Couchbase object by scanning a configured number of rows of the object. You can use RowScanDepth, UseInfer, FlattenArrays, SchemaDepth, SelectArrayIndexes and FlattenObjects to control the relational representation of the documents in Couchbase.

Optionally, you can use Custom Schema Definitions to project your chosen relational structure on top of a Couchbase object. This allows you to define your chosen column names, their data types, and the location of their values in the Couchbase document.

See Query Mapping for more details on how various N1QL operations are represented as SQL.

Automatic Schema Discovery

The adapter automatically infers a relational schema by inspecting a series of Couchbase documents in a collection. You can use the RowScanDepth property to define the number of documents the adapter will scan to do so. The columns identified during the discovery process depend on the FlattenArrays and FlattenObjects properties.

If FlattenObjects is set, all nested objects will be flattened into a series of columns. For example, consider the following document:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
  address: {street: "Main Street", city: "Chapel Hill", state:
"NC"},
  offices: ["Chapel Hill", "London", "New York"],
  annual_revenue: 35,600,000
}
```

This document will be represented by the following columns:

Column Name	Data Type	Example Value
id	Integer	12
name	String	Lohia Manufacturers Inc.
address_street	String	Main Street
address_city	String	Chapel Hill
address_state	String	NC
offices	String	["Chapel Hill", "London", "New York"]
annual_revenue	Double	35,600,000

If FlattenObjects is not set, then the address_string, address_city, and address_state columns will not be broken apart. The address column of type string will instead represent the entire object. Its value would be the following:

```
{street: "Main Street", city: "Chapel Hill", state: "NC"}
```

The FlattenArrays property can be used to flatten array values into columns of their own. This is only recommended for arrays that are expected to be short, for example the coordinates below:

```
"coord": [ -73.856077, 40.848447 ]
```

The FlattenArrays property can be set to 2 to represent the array above as follows:

Column Name	Data Type	Example Value
coord_0	Float	-73.856077
coord_1	Float	40.848447

When UseInfer is set to true, the adapter uses Couchbase's Infer command to detect columns in a bucket. Please note that the datatypes for these columns are still scanned internally, this helps to support a larger set of datatypes not specified using the Infer command. The adapter also gives the user greater control over the Infer command using these additional connection properties: SchemaDepth, InferNumSampleValues, InferSimilarityMetric, and InferSampleSize. It also enables querying certain flavors, and will list columns that exist in that flavor only, assume the following documents in a bucket named Items.

```
{
  "made_in": "New Zealand",
  "amount_in_stock": 200,
  "production_date": "2016-01-10",
  "type": "technology"
  ... Other data
}


{
  "category": "fruits",
  "amount_in_stock": 24,
  "expiration_date": "2017-01-10",
  "type": "food"
  ... Other data
}
```

Infer will detect a few flavors here, two of which would be type=food and type=technology. You can query these flavors with the following queries:

```
SELECT * FROM [Items.Technology]
```

Results:

made_in	amount_in_stock	production_date	type

New Zealand	200	2016-01-10	technology
null	24	null	food

Notice how expiration_date and category columns are ommited here, other flavors can be queried in a similiar fashion:

```
SELECT * FROM [Items.Food]
```

Results:

category	amount_in_stock	expiration_date	type
null	200	null	technology
fruits	24	2017-01-10	food

Setting UseInfer to Infer also allows setting a certain depth to Couchbase objects when converting them to fields using SchemaDepth. For example, consider the following document:

```
{
  "parent": {
    "child1": {
      "child2": {
        "child3": {
          "field": "Test"
        }
      }
    }
  }
}
```

When SchemaDepth set to 2, the query:

```
SELECT * FROM Table
```

Will produce the following result:

parent.child1

```
"child2": {
  "child3": {
    "field": "Test"
  }
}
```

When SchemaDepth set to 2, the same query will produce:

parent.child1.child2

```
"child3": {
  "field": "Test"
}
```

Query Mapping

The adapter maps SQL-92-compliant queries into corresponding N1QL queries. A detailed description of all the transformations is out of scope, but we will describe some of the common elements that are used.

SELECT Queries

The SELECT statements are translated to the appropriate N1QL SELECT query as shown below. Due to the similarities between SQL-92 and N1QL, many queries will simply be direct translations.

One major difference is that when the schema for a given Couchbase bucket exists in the adapter, a SELECT * query will be translated to directly select the individual fields in the bucket. The adapter will also automatically create an Id column based on the primary key of each document in the bucket.

SQL Query	N1QL Query

SELECT * FROM Users	SELECT META(`USERS`).id AS `id`, ... FROM `Users`
SELECT [Document.Id], status FROM Users	SELECT META(`Users`).id AS `Document.Id`, `status` FROM `Users`
SELECT * FROM Users WHERE status = "A"	SELECT META(`USERS`).id AS `id`, ... FROM `Users` WHERE TOSTRING(`Users`.`status`) = "A"
SELECT * FROM Users WHERE status = "A" OR age=50	SELECT META(`USERS`).id AS `id`, ... FROM `Users` WHERE TOSTRING(`Users`.`status`) = "A" OR `Users`.`age`=50
SELECT * FROM Users WHERE name LIKE 'A%'	SELECT META(`USERS`).id AS `id`, ... FROM `Users` WHERE TOSTRING(`Users`.`name`) LIKE "A%"
SELECT * FROM Users WHERE status = "A" ORDER BY [Document.Id] ASC	SELECT META(`USERS`).id AS `id`, ... FROM `Users` WHERE TOSTRING(`Users`.`status`) = "A" ORDER BY META(`Users`).id ASC
SELECT * FROM Users WHERE status = "A" ORDER BY [Document.Id] DESC	SELECT META(`USERS`).id AS `id`, ... FROM `Users` WHERE TOSTRING(`Users`.`status`) = "A" ORDER BY META(`Users`).id DESC

NEST and UNNEST statements

N1QLs NEST and UNNEST statements are also supported, the following queries are supported:

SQL Query	N1QL Query
SELECT *, Address FROM Users UNNEST Addresses AS Address	SELECT *, `Address` FROM `Users` UNNEST `Addresses` AS `Address`

SELECT U.Name, L.Date, U.LoginId FROM USERS U NEST Logins L ON U.LoginId	SELECT `U`.`Name`, `L`.`Date`, `U`.`LoginId` FROM `USERS` AS `U` NEST `Logins` AS `L` ON KEYS `U`.`LoginId`

Aggregate Queries

N1QL has several built-in aggregate functions. The adapter makes extensive use of this for various aggregate queries. See some examples below:

SQL Query	N1QL Query
SELECT Count(*) As Count FROM Orders	SELECT Count(*) AS `count` FROM `Orders`
SELECT Sum(price) As total FROM Orders	SELECT Sum(`price`) As `total` FROM `Orders`
SELECT cust_id, Sum(price) As total FROM Orders GROUP BY cust_id ORDER BY total	SELECT `cust_id`, Sum(`price`) As `total` FROM `Orders` GROUP BY `cust_id` ORDER BY `total`
SELECT cust_id, ord_date, Sum(price) As total FROM Orders GROUP BY cust_id, ord_date Having total > 250	SELECT `cust_id`, `ord_date`, Sum(`price`) As `total` FROM `Orders` GROUP BY `cust_id`, `ord_date` Having `total` > 250

Insert Statements

The SQL INSERT statement is mapped to the N1QL INSERT statement as shown below:

SQL Query	N1QL Query
INSERT INTO users([Document.Id], age, status) VALUES ("bcd001", 45, "A")	INSERT INTO `users` (KEY, VALUE) VALUES ('bcd001', { "age" : 45, "status" : "A" })

Bulk Insert Statements

Bulk inserts are also supported the SQL Bulk Insert is converted as shown below:

```
INSERT INTO Users#Temp([Document.Id], KEY, VALUE) VALUES('bcd001',
45, "A")


INSERT INTO Users#Temp([Document.Id], KEY, VALUE) VALUES('bcd002',
24, "B")


INSERT INTO Users SELECT * FROM Users#Temp
```

is converted to:

```
INSERT INTO `users` (KEY, VALUE)
\tVALUES ('bcd001', { "age" : 45, "status" : "A" })
\tVALUES ('bcd002', { "age" : 24, "status" : "B" })
```

Update Statements

The SQL UPDATE statement is mapped to the N1SQL UPDATE statement as shown below:

SQL Query	N1QL Query
UPDATE users SET status = "C" WHERE [Document.Id] = "bcd001"	UPDATE `users` USE KEYS "bcd001" SET `status`="C"
UPDATE users SET status = "C" WHERE Age > 45	UPDATE `users` SET `status` = "C" WHERE `Age` > 45

Delete Statements

The SQL DELETE statement is mapped to the N1QL DELETE statement as shown below:

SQL Query	N1QL Query
DELETE FROM users WHERE [Document.Id] = "bcd001"	DELETE FROM `users` USE KEYS "bcd001"

DELETE FROM users WHERE status = "inactive"	DELETE FROM `users` WHERE `status` = "inactive"

Custom Schema Definitions

In addition to Automatic Schema Discovery the adapter also allows you to statically define the schema for your Couchbase object. Let's consider the schema below and extract out the nested properties as their own columns:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
  homeaddress: {street: "Main Street", city: "Chapel Hill", state:
"NC"},
  workaddress: {street: "10th Street", city: "Chapel Hill", state:
"NC"}
  offices: ["Chapel Hill", "London", "New York"]
  annual_revenue: 35,600,000
}
{
  id: 15,
  name: "Piago Industries",
  homeaddress: {street: "Main Street", city: "San Francisco",
state: "CA"},
  workaddress: {street: "10th Street", city: "San Francisco",
state: "CA"}
  offices: ["Durham", "San Francisco"]
  annual_revenue: 42,600,000
}
```

Custom Schema Definition

```
<rsb:info title="Customers" description="Customers">
  <!-- Column definitions -->
    <attr name="Id"                 xs:type="integer" key="true"
other:defaultAlias="id" />
    <attr name="Name"               xs:type="string"
other:defaultAlias="name" />
    <attr name="HomeAddress_Street" xs:type="string"
other:defaultAlias="street" />
    <attr name="HomeAddress_City"   xs:type="string"
other:defaultAlias="city" />
    <attr name="WorkAddress_Street" xs:type="string"
other:defaultAlias="street1" />
    <attr name="WorkAddress_City"   xs:type="string"
other:defaultAlias="city1" />
    <attr name="Offices"            xs:type="string"
other:defaultAlias="offices" />
</rsb:info>
```

The schema above uses the following properties to define specific qualities for each column:

Property	Meaning
other:defaultAlias	Designates the default alias to use in the N1QL query. This is important when you have multiple columns with the same leaf name. Each defaultAlias should be unique.

In Custom Schema Example, you will find the complete schema that contains the example above.

Custom Schema Example

In this section is a complete schema. The info section enables a relational view of a Couchbase object. For more details, see Custom Schema Definitions. The table below allows the SELECT, INSERT, UPDATE, and DELETE commands as implemented in the GET, POST, MERGE, and DELETE sections of the schema below. The operations, such as couchbaseadoSysData, are internal implementations.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

  <rsb:info title="Customers" description="Customers">
    <!-- Column definitions -->
    <attr name="Id"                 xs:type="integer" key="true"
other:defaultAlias="id"      />
    <attr name="Name"               xs:type="string"
other:defaultAlias="name"    />
    <attr name="HomeAddress_Street" xs:type="string"
other:defaultAlias="street"  />
    <attr name="HomeAddress_City"   xs:type="string"
other:defaultAlias="city"    />
    <attr name="WorkAddress_Street" xs:type="string"
other:defaultAlias="street1" />
    <attr name="WorkAddress_City"   xs:type="string"
other:defaultAlias="city1"   />
    <attr name="Offices"            xs:type="string"
other:defaultAlias="offices" />
  </rsb:info>

  <rsb:set attr="bucket" value="customers"/>

  <rsb:script method="GET">
    <rsb:push op="couchbaseadoSysData" input="_input"/>
  </rsb:script>
```

```
  <rsb:script method="POST">
    <rsb:push op="couchbaseadoSysData" input="_input"/>
  </rsb:script>

  <rsb:script method="MERGE">
    <rsb:push op="couchbaseadoSysData" input="_input"/>
  </rsb:script>

  <rsb:script method="DELETE">
    <rsb:push op="couchbaseadoSysData" input="_input"/>
  </rsb:script>

</rsb:script>
```

SQL Compliance

The Couchbase Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Couchbase API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

CREATE TABLE Statements

See CREATE TABLE Statements for a syntax reference and examples.

DROP TABLE Statements

See DROP TABLE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Couchbase adapter:

```
SELECT {
  [ TOP <numeric_literal> | DISTINCT ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  } [ , ... ]
  [ [
      INNER | { { LEFT | RIGHT | FULL  } [ OUTER ] }
    ] JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [ HAVING <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
     <expression> { = | != | < | <= | > | >= | IS NULL | LIKE | AND
| OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Customer
```

Rename a column:
```
SELECT "TotalDue" AS MY_TotalDue FROM Customer
```

Search data:
```
SELECT * FROM Customer WHERE CustomerId = '12345';
```

The Couchbase APIs support the following operators in the WHERE clause: =, !=, <, <=, >, >=, IS NULL, LIKE, AND, OR.
```
SELECT * FROM Customer WHERE CustomerId = '12345';
```

Return the number of items matching the query criteria:
```
SELECT COUNT(*) AS MyCount FROM Customer
```

Return the unique items matching the query criteria:
```
SELECT DISTINCT TotalDue FROM Customer
```

Summarize data:
```
SELECT TotalDue, MAX(AnnualRevenue) FROM Customer  GROUP BY
TotalDue
```

See Aggregate Functions for details.

Retrieve data from multiple tables.
```
SELECT Orders.OrderDate, Customers.ContactName FROM Customers
INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
```

See JOIN Queries for details.

Sort a result set in ascending order:
```
SELECT Name, TotalDue FROM Customer  ORDER BY TotalDue ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.
```
SELECT COUNT(*) FROM Customer WHERE CustomerId = '12345'
```

COUNT(DISTINCT)

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT(DISTINCT Name) AS DistinctValues FROM Customer WHERE
CustomerId = '12345'
```

AVG

Returns the average of the column values.

```
SELECT TotalDue, AVG(AnnualRevenue) FROM Customer WHERE CustomerId
= '12345' GROUP BY TotalDue
```

MIN

Returns the minimum column value.

```
SELECT MIN(AnnualRevenue), TotalDue FROM Customer WHERE CustomerId
= '12345' GROUP BY TotalDue
```

MAX

Returns the maximum column value.

```
SELECT TotalDue, MAX(AnnualRevenue) FROM Customer WHERE CustomerId
= '12345' GROUP BY TotalDue
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(AnnualRevenue) FROM Customer WHERE CustomerId = '12345'
```

JOIN Queries

In Couchbase, documents can be joined if they share a common key. Only inner join is supported.

The following examples use tables in the Northwind database to show how to join Couchbase tables:

The query below returns the ContactName and the OrderDate of every customer who has an order:

```
SELECT Customers.ContactName, Orders.OrderDate
FROM Orders, Customers
WHERE Customers.CustomerID=Orders.CustomerID
```

You can use the implicit syntax as well as the INNER JOIN keyword:

```
SELECT Orders.OrderDate, Customers.ContactName
FROM Customers
INNER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
```

Projection Functions

ARRAY_AGG(column)

Returns array of the non-MISSING values in the group, including NULL values.

column: Any column expression.

ARRAY_APPEND(column, value)

Returns new array with value appended.

column: Any column expression.

value: The value to be appended to the array.

ARRAY_CONCAT(column1, column2)

Returns new array with the concatenation of the input arrays.

column1: Any column expression.

column2: Any column expression.

ARRAY_DISTINCT(column)

Returns new array with distinct elements of input array.

column: Any column expression.

ARRAY_IFNULL(column)

Returns the first non-NULL value in the array, or NULL.

column: Any column expression.

ARRAY_PREPEND(column, value)

Returns new array with value pre-pended.

column: Any column expression.

value: The value to be pre-pended to the array.

ARRAY_PUT(column, value)

Returns new array with value appended, if value is not already present, otherwise returns the unmodified input array.

column: Any column expression.

value: The value to append to the array.

ARRAY_REMOVE(column, value)

Returns new array with all occurrences of value removed.

column: Any column expression.

value: The value to be removed from the array.

ARRAY_REPLACE(column, value1, value2 [, integer_n])

Returns new array with all occurrences of value removed.

column: Any column expression.

value1: The value to be replaced by value2.

value2: The value to replace value1.

integer_n: The maximum number of replacements to be performed.

ARRAY_REVERSE(column)

Returns new array with all elements in reverse order.

column: Any column expression.

ARRAY_SORT(column)

Returns new array with elements sorted in N1QL collation order.

column: Any column expression.

DECODE_JSON(column)

Unmarshals the JSON-encoded string into a N1QL value. The empty string is MISSING.

column: Any column expression.

ENCODE_JSON(column)

Marshals the N1QL value into a JSON-encoded string. MISSING becomes the empty string.

column: Any column expression.

ENCODED_SIZE(column)

Number of bytes in an uncompressed JSON encoding of the value. The exact size is implementation-dependent. Always returns an integer, and never MISSING or NULL. Returns 0 for MISSING.

column: Any column expression.

POLY_LENGTH(column)

Returns length of the value after evaluating the expression. The exact meaning of length depends on the type of the value: MISSING: MISSING; NULL: NULL; String: The length of the string.; Array: The number of elements in the array.; Object: The number of name/value pairs in the object; Any other value: NULL.

column: Any column expression.

OBJECT_LENGTH(column)

Returns number of name-value pairs in the object.

column: Any column expression.

OBJECT_NAMES(column)

Returns array containing the attribute names of the object, in N1QL collation order.

column: Any column expression.

OBJECT_PAIRS(column)

Returns array containing the attribute name and value pairs of the object, in N1QL collation order of the names.

column: Any column expression.

OBJECT_VALUES(column)

Returns array containing the attribute values of the object, in N1QL collation order of the corresponding names.

column: Any column expression.

ARRAY_AVG(column)

Returns arithmetic mean (average) of all the non-NULL number values in the array, or NULL if there are no such values.

column: Any column expression.

ARRAY_CONTAINS(column, value)

Returns true if the array contains value.

column: Any column expression.

value: The value contained within the array.

ARRAY_COUNT(column)

Returns count of all the non-NULL values in the array, or zero if there are no such values.

column: Any column expression.

ARRAY_LENGTH(column)

Returns the number of elements in the array.

column: Any column expression.

ARRAY_MAX(column)

Returns the largest non-NULL, non-MISSING array element, in N1QL collation order.

column: Any column expression.

ARRAY_MIN(column)

Returns smallest non-NULL, non-MISSING array element, in N1QL collation order.

column: Any column expression.

ARRAY_POSITION(column, value)

Returns the first position of value within the array, or -1. Array position is zero-based, i.e. the first position is 0.

column: Any column expression.

value: The value contained within the array.

ARRAY_SUM(column)

Sum of all the non-NULL number values in the array, or zero if there are no such values.

column: Any column expression.

GREATEST(column1, column2 [,column3 [,column4]])

Largest non-NULL, non-MISSING value if the values are of the same type; otherwise NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

LEAST(column1, column2 [,column3 [,column4]])

Returns smallest non-NULL, non-MISSING value if the values are of the same type, otherwise returns NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFMISSING(column1, column2 [,column3 [,column4]])

Returns the first non-MISSING value.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFMISSINGORNULL(column1, column2 [,column3 [,column4]])

Returns first non-NULL, non-MISSING value.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFNULL(column1, column2 [,column3 [,column4]])

Returns first non-NULL value. Note that this function might return MISSING if there is no non-NULL value.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

MISSINGIF(column1, column2)

Returns MISSING if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

NULLIF(column1, column2)

Returns NULL if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

IFINF(column1, column2 [,column3 [,column4]])

Returns first non-MISSING, non-Inf number. Returns MISSING or NULL if a non-number input is encountered first.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFNAN(column1, column2 [,column3 [,column4]])

Returns first non-MISSING, non-NaN number. Returns MISSING or NULL if a non-number input is encountered first.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFNANORINF(column1, column2 [,column3 [,column4]])

Returns first non-MISSING, non-Inf, or non-NaN number. Returns MISSING or NULL if a non-number input is encountered first.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

NANIF(column1, column2 [,column3 [,column4]])

Returns NaN if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

NEGINFIF(column1, column2 [,column3 [,column4]])

Returns NegInf if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

POSINFIF(column1, column2 [,column3 [,column4]])

Returns PosInf if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

CLOCK_MILLIS()

Returns system clock at function evaluation time, as UNIX milliseconds. Varies during a query.

CLOCK_STR([string_fmt])

Returns system clock at function evaluation time, as a string in a supported format. Varies during a query.

string_fmt: The datetime format to return the system clock in.

DATE_ADD_MILLIS(column, integer_n, string_part)

Performs date arithmetic, and returns result of computation. n and part are used to define an interval or duration, which is then added (or subtracted) to the UNIX time stamp, returning the result.

column: Any column expression.

integer_n: The number of string_part's to add to the column value.

string_part: The part to add integer_n to, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_ADD_STR(column, integer_n, string_part)

Performs date arithmetic. n and part are used to define an interval or duration, which is then added (or subtracted) to the date string in a supported format, returning the result.

column: Any column expression.

integer_n: The number of string_part's to add to the column value.

string_part: The part to add integer_n to, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_DIFF_MILLIS(column1, column2, string_part)

Performs date arithmetic. Returns the elapsed time between two UNIX time stamps as an integer whose unit is part.

column1: Any column expression.

column2: Any column expression.

string_part: The unit of the result, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_DIFF_STR(column1, column2, string_part)

Performs date arithmetic. Returns the elapsed time between two date strings in a supported format, as an integer whose unit is part.

column1: Any column expression.

column2: Any column expression.

string_part: The unit of the result, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_PART_MILLIS(column1, string_part [, tz])

Returns date part as an integer. The date expression is a number representing UNIX milliseconds, and part is one of the following date part strings.

column1: Any column expression.

string_part: The component of the date to extract. Available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, millisecond, day_of_year, day_of_week, iso_week, iso_year, iso_dow, timezone, timezone_hour, and timezone_minute.

tz: The timezone to convert the local time to. Default to the system timezone if not specified. If an incorrect time zone is provided, the null is returned.

DATE_PART_STR(column1, string_part)

Returns date part as an integer. The date expression is a string in a supported format, and part is one of the supported date part strings.

column1: Any column expression.

string_part: The unit of the result, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, millisecond, day_of_year, day_of_week, iso_week, iso_year, iso_dow, timezone, timezone_hour, and timezone_minute.

DATE_TRUNC_MILLIS(column1, string_part)

Returns UNIX time stamp that has been truncated so that the given date part string is the least significant.

column1: Any column expression.

string_part: The least significant date part, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_TRUNC_STR(column1, string_part)

Returns ISO 8601 time stamp that has been truncated so that the given date part string is the least significant.

column1: Any column expression.

string_part: The least significant date part, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

MILLIS(column1)

Returns date that has been converted in a supported format to UNIX milliseconds.

column1: Any column expression.

STR_TO_MILLIS(column1)

Returns date that has been converted in a supported format to UNIX milliseconds.

column1: Any column expression.

MILLIS_TO_STR(column [, string_fmt])

Returns the string in the supported format to which the UNIX milliseconds has been converted.

column1: Any column expression.

string_fmt: The datetime format to return the system clock in.

MILLIS_TO_UTC(column [, string_fmt])

Returns the UTC string to which the UNIX time stamp has been converted in the supported format.

column1: Any column expression.

string_fmt: The datetime format to return the system clock in.

MILLIS_TO_TZ(column, string_tzname [, string_fmt])

Converts the UNIX time stamp to a string in the named time zone, and returns the string.

column1: Any column expression.

string_tzname: The time zone name.

string_fmt: The datetime format to return the system clock in.

NOW_MILLIS()

Returns statement time stamp as UNIX milliseconds; does not vary during a query.

NOW_STR([string_fmt])

Returns statement time stamp as a string in a supported format; does not vary during a query.

string_fmt: The datetime format to return the timestamp in.

STR_TO_UTC(column1)

Converts the ISO 8601 time stamp to UTC.

column1: Any column expression.

STR_TO_ZONE_NAME(column, string_tzname)

Converts the supported time stamp string to the named time zone.

column1: Any column expression.

string_tzname: The time zone name.

BASE64(expression)

Returns base64 encoding of expression.

expression: Any column or literal expression.

ABS(expression)

Returns absolute value of the number.

expression: Any column or literal expression.

ACOS(expression)

Returns arccosine in radians.

expression: Any column or literal expression.

ASIN(expression)

Returns arcsine in radians.

expression: Any column or literal expression.

ATAN(expression)

Returns arctangent in radians.

expression: Any column or literal expression.

ATAN2(expression1, expression2)

Returns arctangent of expression2/expression1.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

CEIL(expression)

Returns smallest integer not less than the number.

expression: Any column or literal expression.

COS(expression)

Returns cosine.

expression: Any column or literal expression.

DEGREES(expression)

Returns radians to degrees.

expression: Any column or literal expression.

E()

Base of natural logarithms.

EXP(expression)

Returns e^expression.

expression: Any column or literal expression.

LN(expression)

Returns log base e.

expression: Any column or literal expression.

LOG(expression)

Returns log base 10.

expression: Any column or literal expression.

FLOOR(expression)

Largest integer not greater than the number.

expression: Any column or literal expression.

PI()

Returns PI.

POWER(expression1, expression2)

Returns expression1^expression2.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

RADIANS(expression)

Returns degrees to radians.

expression: Any column or literal expression.

RANDOM([expression])

Returns pseudo-random number with optional seed.

expression: Any column or literal expression.

ROUND(expression [, integer_digits])

Rounds the value to the given number of integer digits to the right of the decimal point (left if digits is negative). Digits is 0 if not given.

expression: Any column or literal expression.

integer_digits: The number of digits to round to.

SIGN(expression)

Valid values: -1, 0, or 1 for negative, zero, or positive numbers respectively.

expression: Any column or literal expression.

SIN(expression)

Returns sine.

expression: Any column or literal expression.

SQRT(expression)

Returns square root.

expression: Any column or literal expression.

TAN(expression)

Returns tangent.

expression: Any column or literal expression.

TRUNC(expression [, integer_digits])

Truncates the number to the given number of integer digits to the right of the decimal point (left if digits is negative). Digits is 0 if not given.

expression: Any column or literal expression.

integer_digits: The number of digits to truncate.

CONTAINS(column, string_substring)

True if the string contains the substring.

column: Any column or literal expression.

string_substring: The substring to search for.

INITCAP(column)

Converts the string so that the first letter of each word is uppercase and every other letter is lowercase.

column: Any column or literal expression.

LENGTH(column)

Returns length of the string value.

column: Any column or literal expression.

LOWER(column)

Returns lowercase of the string value.

column: Any column or literal expression.

LTRIM(column [, string_chars])

Returns string with all leading chars removed. White space by default.

column: Any column or literal expression.

string_chars: The leading characters to remove.

POSITION(column, string_substring)

Returns the first position of the substring within the string, or -1. The position is zero-based, i.e., the first position is 0.

column: Any column or literal expression.

string_substring: The substring to search for.

REPEAT(column, integer_n)

Returns string formed by repeating expression n times.

column: Any column or literal expression.

integer_n: The number of times to repeat column.

REPLACE(column, string_substring, string_replace [, integer_n])

Returns string with all occurrences of substr replaced with repl. If n is given, at most n replacements are performed.

column: The column expression.

string_substring: The regular expression to match.

string_replace: The value to replace the matched pattern.

integer_n: The maximum number of replacements to make.

RTRIM(column [, string_chars])

Returns string with all trailing chars removed. White space by default.

column: Any column or literal expression.

string_chars: The trailing characters to remove.

SPLIT(column [, string_sep])

Splits the string into an array of substrings separated by string_sep. If string_sep is not given, any combination of white space characters is used.

column: Any column or literal expression.

string_sep: The separator to split column on.

SUBSTR(column, integer_position [, integer_length])

Returns substring from the integer position of the given length, or to the end of the string. The position is zero-based, i.e. the first position is 0. If position is negative, it is counted from the end of the string; -1 is the last position in the string.

column: Any column or literal expression.

integer_position: The starting position.

integer_length: The total length of the substring to retrieve.

TRIM(column [, string_chars])

Returns string with all leading and trailing chars removed. White space by default.

column: Any column or literal expression.

string_chars: The leading and trailing characters to remove.

UPPER(column)

Returns uppercase of the string value.

column: Any column or literal expression.

TOARRAY(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; Arrays are themselves; All other values are wrapped in an array.

column: Any column expression.

TOATOM(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; Arrays of length 1 are the result of TOATOM() on their single element; Objects of length 1 are the result of TOATOM() on their single value; Booleans, numbers, and strings are themselves; All other values are NULL.

column: Any column expression.

TOBOOLEAN(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; False is false; Numbers +0, -0, and NaN are false; Empty strings, arrays, and objects are false; All other values are true.

column: Any column expression.

TONUMBER(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; False is 0; True is 1; Numbers are themselves; Strings that parse as numbers are those numbers; All other values are NULL.

column: Any column expression.

TOOBJECT(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; Objects are themselves; All other values are the empty object.

column: Any column expression.

TOSTRING(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; False is "false"; True is "true"; Numbers are their string representation; Strings are themselves; All other values are NULL.

column: Any column expression.

Predicate Functions

REGEXP_CONTAINS(column, string_pattern)

Returns True if the string value contains the regular expression pattern.

column: The column expression.

string_pattern: The regular expression to match.

REGEXP_LIKE(column, string_pattern)

Returns True if the string value matches the regular expression pattern.

column: The column expression.

string_pattern: The regular expression to match.

REGEXP_POSITION(column, string_pattern)

Returns first position of the regular expression pattern within the string, or -1.

column: The column expression.

string_pattern: The regular expression to match.

REGEXP_REPLACE(column, string_pattern, string_replace [, integer_n])

Returns new string with occurrences of pattern replaced with string_replace. If n is given, at most n replacements are performed.

column: The column expression.

string_pattern: The regular expression to match.

string_replace: The value to replace the matched pattern.

integer_n: The maximum number of replacements to make.

ISARRAY(column)

Returns True if expression is an array, otherwise returns MISSING, NULL or false.

column: Any column expression.

ISATOM(column)

Returns True if expression is a Boolean, number, or string, otherwise returns MISSING, NULL or false.

column: Any column expression.

ISBOOLEAN(column)

Returns True if expression is a Boolean, otherwise returns MISSING, NULL or false.

column: Any column expression.

ISNUMBER(column)

Returns True if expression is a number, otherwise returns MISSING, NULL or false.

column: Any column expression.

ISOBJECT(column)

Returns True if expression is an object, otherwise returns MISSING, NULL or false.

column: Any column expression.

ISSTRING(column)

Returns True if expression is a string, otherwise returns MISSING, NULL or false.

column: Any column expression.

TYPE(column)

Returns one of the following strings, based on the value of expression: missing, null, boolean, number, string, array, object, or binary.

column: Any column expression.

ARRAY_AVG(column)

Returns arithmetic mean (average) of all the non-NULL number values in the array, or NULL if there are no such values.

column: Any column expression.

ARRAY_CONTAINS(column, value)

Returns true if the array contains value.

column: Any column expression.

value: The value contained within the array.

ARRAY_COUNT(column)

Returns count of all the non-NULL values in the array, or zero if there are no such values.

column: Any column expression.

ARRAY_LENGTH(column)

Returns the number of elements in the array.

column: Any column expression.

ARRAY_MAX(column)

Returns the largest non-NULL, non-MISSING array element, in N1QL collation order.

column: Any column expression.

ARRAY_MIN(column)

Returns smallest non-NULL, non-MISSING array element, in N1QL collation order.

column: Any column expression.

ARRAY_POSITION(column, value)

Returns the first position of value within the array, or -1. Array position is zero-based, i.e. the first position is 0.

column: Any column expression.

value: The value contained within the array.

ARRAY_SUM(column)

Sum of all the non-NULL number values in the array, or zero if there are no such values.

column: Any column expression.

GREATEST(column1, column2 [,column3 [,column4]])

Largest non-NULL, non-MISSING value if the values are of the same type; otherwise NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

LEAST(column1, column2 [,column3 [,column4]])

Returns smallest non-NULL, non-MISSING value if the values are of the same type, otherwise returns NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFMISSING(column1, column2 [,column3 [,column4]])

Returns the first non-MISSING value.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFMISSINGORNULL(column1, column2 [,column3 [,column4]])

Returns first non-NULL, non-MISSING value.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFNULL(column1, column2 [,column3 [,column4]])

Returns first non-NULL value. Note that this function might return MISSING if there is no non-NULL value.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

MISSINGIF(column1, column2)

Returns MISSING if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

NULLIF(column1, column2)

Returns NULL if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

IFINF(column1, column2 [,column3 [,column4]])

Returns first non-MISSING, non-Inf number. Returns MISSING or NULL if a non-number input is encountered first.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFNAN(column1, column2 [,column3 [,column4]])

Returns first non-MISSING, non-NaN number. Returns MISSING or NULL if a non-number input is encountered first.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

IFNANORINF(column1, column2 [,column3 [,column4]])

Returns first non-MISSING, non-Inf, or non-NaN number. Returns MISSING or NULL if a non-number input is encountered first.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

NANIF(column1, column2 [,column3 [,column4]])

Returns NaN if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

NEGINFIF(column1, column2 [,column3 [,column4]])

Returns NegInf if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

POSINFIF(column1, column2 [,column3 [,column4]])

Returns PosInf if column1 = column2, otherwise returns column1. Returns MISSING or NULL if either input is MISSING or NULL.

column1: Any column expression.

column2: Any column expression.

column3: Any column expression.

column4: Any column expression.

CLOCK_MILLIS()

Returns system clock at function evaluation time, as UNIX milliseconds. Varies during a query.

CLOCK_STR([string_fmt])

Returns system clock at function evaluation time, as a string in a supported format. Varies during a query.

string_fmt: The datetime format to return the system clock in.

DATE_ADD_MILLIS(column, integer_n, string_part)

Performs date arithmetic, and returns result of computation. n and part are used to define an interval or duration, which is then added (or subtracted) to the UNIX time stamp, returning the result.

column: Any column expression.

integer_n: The number of string_part's to add to the column value.

string_part: The part to add integer_n to, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_ADD_STR(column, integer_n, string_part)

Performs date arithmetic. n and part are used to define an interval or duration, which is then added (or subtracted) to the date string in a supported format, returning the result.

column: Any column expression.

integer_n: The number of string_part's to add to the column value.

string_part: The part to add integer_n to, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_DIFF_MILLIS(column1, column2, string_part)

Performs date arithmetic. Returns the elapsed time between two UNIX time stamps as an integer whose unit is part.

column1: Any column expression.

column2: Any column expression.

string_part: The unit of the result, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_DIFF_STR(column1, column2, string_part)

Performs date arithmetic. Returns the elapsed time between two date strings in a supported format, as an integer whose unit is part.

column1: Any column expression.

column2: Any column expression.

string_part: The unit of the result, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_PART_MILLIS(column1, string_part [, tz])

Returns date part as an integer. The date expression is a number representing UNIX milliseconds, and part is one of the following date part strings.

column1: Any column expression.

string_part: The component of the date to extract. Available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, millisecond, day_of_year, day_of_week, iso_week, iso_year, iso_dow, timezone, timezone_hour, and timezone_minute.

tz: The timezone to convert the local time to. Default to the system timezone if not specified. If an incorrect time zone is provided, the null is returned.

DATE_PART_STR(column1, string_part)

Returns date part as an integer. The date expression is a string in a supported format, and part is one of the supported date part strings.

column1: Any column expression.

string_part: The unit of the result, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, millisecond, day_of_year, day_of_week, iso_week, iso_year, iso_dow, timezone, timezone_hour, and timezone_minute.

DATE_TRUNC_MILLIS(column1, string_part)

Returns UNIX time stamp that has been truncated so that the given date part string is the least significant.

column1: Any column expression.

string_part: The least significant date part, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

DATE_TRUNC_STR(column1, string_part)

Returns ISO 8601 time stamp that has been truncated so that the given date part string is the least significant.

column1: Any column expression.

string_part: The least significant date part, available values are: millennium, century, decade, year, quarter, month, week, day, hour, minute, second, and millisecond.

MILLIS(column1)

Returns date that has been converted in a supported format to UNIX milliseconds.

column1: Any column expression.

STR_TO_MILLIS(column1)

Returns date that has been converted in a supported format to UNIX milliseconds.

column1: Any column expression.

MILLIS_TO_STR(column [, string_fmt])

Returns the string in the supported format to which the UNIX milliseconds has been converted.

column1: Any column expression.

string_fmt: The datetime format to return the system clock in.

MILLIS_TO_UTC(column [, string_fmt])

Returns the UTC string to which the UNIX time stamp has been converted in the supported format.

column1: Any column expression.

string_fmt: The datetime format to return the system clock in.

MILLIS_TO_TZ(column, string_tzname [, string_fmt])

Converts the UNIX time stamp to a string in the named time zone, and returns the string.

column1: Any column expression.

string_tzname: The time zone name.

string_fmt: The datetime format to return the system clock in.

NOW_MILLIS()

Returns statement time stamp as UNIX milliseconds; does not vary during a query.

NOW_STR([string_fmt])

Returns statement time stamp as a string in a supported format; does not vary during a query.

string_fmt: The datetime format to return the timestamp in.

STR_TO_UTC(column1)

Converts the ISO 8601 time stamp to UTC.

column1: Any column expression.

STR_TO_ZONE_NAME(column, string_tzname)

Converts the supported time stamp string to the named time zone.

column1: Any column expression.

string_tzname: The time zone name.

BASE64(expression)

Returns base64 encoding of expression.

expression: Any column or literal expression.

ABS(expression)

Returns absolute value of the number.

expression: Any column or literal expression.

ACOS(expression)

Returns arccosine in radians.

expression: Any column or literal expression.

ASIN(expression)

Returns arcsine in radians.

expression: Any column or literal expression.

ATAN(expression)

Returns arctangent in radians.

expression: Any column or literal expression.

ATAN2(expression1, expression2)

Returns arctangent of expression2/expression1.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

CEIL(expression)

Returns smallest integer not less than the number.

expression: Any column or literal expression.

COS(expression)

Returns cosine.

expression: Any column or literal expression.

DEGREES(expression)

Returns radians to degrees.

expression: Any column or literal expression.

E()

Base of natural logarithms.

EXP(expression)

Returns e^expression.

expression: Any column or literal expression.

LN(expression)

Returns log base e.

expression: Any column or literal expression.

LOG(expression)

Returns log base 10.

expression: Any column or literal expression.

FLOOR(expression)

Largest integer not greater than the number.

expression: Any column or literal expression.

PI()

Returns PI.

POWER(expression1, expression2)

Returns expression1^expression2.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

RADIANS(expression)

Returns degrees to radians.

expression: Any column or literal expression.

RANDOM([expression])

Returns pseudo-random number with optional seed.

expression: Any column or literal expression.

ROUND(expression [, integer_digits])

Rounds the value to the given number of integer digits to the right of the decimal point (left if digits is negative). Digits is 0 if not given.

expression: Any column or literal expression.

integer_digits: The number of digits to round to.

SIGN(expression)

Valid values: -1, 0, or 1 for negative, zero, or positive numbers respectively.

expression: Any column or literal expression.

SIN(expression)

Returns sine.

expression: Any column or literal expression.

SQRT(expression)

Returns square root.

expression: Any column or literal expression.

TAN(expression)

Returns tangent.

expression: Any column or literal expression.

TRUNC(expression [, integer_digits])

Truncates the number to the given number of integer digits to the right of the decimal point (left if digits is negative). Digits is 0 if not given.

expression: Any column or literal expression.

integer_digits: The number of digits to truncate.

CONTAINS(column, string_substring)

True if the string contains the substring.

column: Any column or literal expression.

string_substring: The substring to search for.

INITCAP(column)

Converts the string so that the first letter of each word is uppercase and every other letter is lowercase.

column: Any column or literal expression.

LENGTH(column)

Returns length of the string value.

column: Any column or literal expression.

LOWER(column)

Returns lowercase of the string value.

column: Any column or literal expression.

LTRIM(column [, string_chars])

Returns string with all leading chars removed. White space by default.

column: Any column or literal expression.

string_chars: The leading characters to remove.

POSITION(column, string_substring)

Returns the first position of the substring within the string, or -1. The position is zero-based, i.e., the first position is 0.

column: Any column or literal expression.

string_substring: The substring to search for.

REPEAT(column, integer_n)

Returns string formed by repeating expression n times.

column: Any column or literal expression.

integer_n: The number of times to repeat column.

REPLACE(column, string_substring, string_replace [, integer_n])

Returns string with all occurrences of substr replaced with repl. If n is given, at most n replacements are performed.

column: The column expression.

string_substring: The regular expression to match.

string_replace: The value to replace the matched pattern.

integer_n: The maximum number of replacements to make.

RTRIM(column [, string_chars])

Returns string with all trailing chars removed. White space by default.

column: Any column or literal expression.

string_chars: The trailing characters to remove.

SPLIT(column [, string_sep])

Splits the string into an array of substrings separated by string_sep. If string_sep is not given, any combination of white space characters is used.

column: Any column or literal expression.

string_sep: The separator to split column on.

SUBSTR(column, integer_position [, integer_length])

Returns substring from the integer position of the given length, or to the end of the string. The position is zero-based, i.e. the first position is 0. If position is negative, it is counted from the end of the string; -1 is the last position in the string.

column: Any column or literal expression.

integer_position: The starting position.

integer_length: The total length of the substring to retrieve.

TRIM(column [, string_chars])

Returns string with all leading and trailing chars removed. White space by default.

column: Any column or literal expression.

string_chars: The leading and trailing characters to remove.

UPPER(column)

Returns uppercase of the string value.

column: Any column or literal expression.

TOARRAY(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; Arrays are themselves; All other values are wrapped in an array.

column: Any column expression.

TOATOM(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; Arrays of length 1 are the result of TOATOM() on their single element; Objects of length 1 are the result of TOATOM() on their single value; Booleans, numbers, and strings are themselves; All other values are NULL.

column: Any column expression.

TOBOOLEAN(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; False is false; Numbers +0, -0, and NaN are false; Empty strings, arrays, and objects are false; All other values are true.

column: Any column expression.

TONUMBER(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; False is 0; True is 1; Numbers are themselves; Strings that parse as numbers are those numbers; All other values are NULL.

column: Any column expression.

TOOBJECT(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; Objects are themselves; All other values are the empty object.

column: Any column expression.

TOSTRING(column)

Returns array as follows: MISSING is MISSING; NULL is NULL; False is "false"; True is "true"; Numbers are their string representation; Strings are themselves; All other values are NULL.

column: Any column expression.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Name, TotalDue INTO "csv://Customer.txt" FROM "Customer"
WHERE CustomerId = '12345'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Name, TotalDue INTO "csv://Customer.txt;delimiter=tab" FROM
"Customer" WHERE CustomerId = '12345'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Customer (TotalDue) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Customer SET TotalDue='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:couchbase:User='myusername';Pass
word='mypassword';Server='http://couchbase40'",);
String cmd = "DELETE FROM Customer WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

CREATE TABLE Statements

To create new Couchbase entities, use CREATE TABLE statements:

CREATE TABLE Syntax

The CREATE TABLE statement specifies the table name and a comma-separated list of column names and the primary keys of the table.

```
CREATE TABLE <table_name> IF [ NOT EXISTS ]
(
  {
     <column_name> <data_type>
     [ NOT NULL ]
     [ DEFAULT <literal> ]
     [ PRIMARY KEY ]
     [ UNIQUE ]
  } |  PRIMARY KEY ( <column_name> [ , ... ] )
  [ , ... ]
)
```

Example Query:

The following statement creates a MyCustomers table on the Couchbase server with name, age, and address columns.

```
CREATE TABLE IF NOT EXISTS "MyCustomers" (name VARCHAR(20), age
INT, address VARCHAR(20))
```

DROP TABLE Statements

Use DROP TABLE statements to delete a table and all the data it contains from Couchbase.

DROP TABLE Syntax

The DROP TABLE statement accepts the name of the table to delete.

```
DROP TABLE [ IF EXISTS ] <table_name>
```

Example Query:

The following query deletes all MyCustomers data from the server.

```
DROP TABLE IF EXISTS MyCustomers
```

Data Model

TDV Cassandra Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to the Server

Set the Server and Database connection properties to connect to Cassandra.

To connect to a distributed system, you can set Server to a comma-separated list of servers and ports, separated by colons. You will also need to set ConsistencyLevel.

The following sections detail connection properties for authentication, security, and data access. See Advanced Settings if you need more control over connecting to your system.

Securing Cassandra Connections

You can set UseSSL to negotiate SSL/TLS encryption when you connect. By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Authenticating to Cassandra

The adapter supports Basic authentication with login credentials and the additional authentication features of DataStax Enterprise (DSE) Cassandra. The following sections detail connection properties your authentication method may require.

You need to set AuthScheme to the value corresponding to the authenticator configured for your system. You specify the authenticator in the *authenticator* property in the cassandra.yaml file. This file is typically found in /etc/dse/cassandra. or through the DSE Unified Authenticator on DSE Cassandra.

Basic Authentication

Basic authentication is supported through Cassandra's built-in default PasswordAuthenticator.

Set the AuthScheme property to 'BASIC' and set the User and Password properties.

In the cassandra.yaml file, set the *authenticator* property to 'PasswordAuthenticator'.

Kerberos Authentication

Kerberos authentication is supported through DataStax Enterprise Unified Authentication.

Set the AuthScheme property to 'KERBEROS' and set the User and Password properties.

Set the KerberosKDC, KerberosRealm, and KerberosSPN properties.

In the cassandra.yaml file, set the *authenticator* property to "com.datastax.bdp.cassandra.auth.DseAuthenticator".

Modify the *authentication_options* section in the dse.yaml file, specifying the default_schema and other_schemas properties as 'kerberos'.

Modify the *kerberos_options* section in the dse.yaml file, specifying the keytab, service_principle, http_principle and qop properties

LDAP Authentication

LDAP authentication is supported through DataStax Enterprise Unified Authentication.

Set the AuthScheme property to 'LDAP' and set the User and Password properties.

In the cassandra.yaml file, set the *authenticator* property to "com.datastax.bdp.cassandra.auth.DseAuthenticator".

Modify the *authentication_options* section in the dse.yaml file, specifying the default_schema and other_schemas properties as 'ldap'.

Modify the *ldap_options* section in the dse.yaml file, specifying the server_host, server_port, search_dn, search_password, user_search_base, and user_search_filter properties

Using PKI

You can specify a client certificate to authenticate the adapter with SSLClientCert, SSLClientCertType, SSLClientCertSubject, and SSLClientCertPassword.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Allow Filtering	When true, slow-performing queries are processed on the server.
Auth Scheme	The scheme used for authentication. Accepted entries are BASIC, DSE, KERBEROS, and LDAP.
Case Sensitivity	Enable case sensitivity to the CQL sending to the server, if set to True, the identifiers in the CQL will be enclosed in double quotation marks.
Consistency Level	The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success.
Database	The name of the Cassandra keyspace.
Default LDAP User	The default LDAP user used to connect to and communicate with the server, it must be set if the LDAP server do not allow anonymous bind.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.

Firewall User	The user name to use to authenticate with a proxy-based firewall.
Flatten Arrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
Flatten Objects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
Kerberos KDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
Kerberos Realm	The Kerberos Realm used to authenticate the user with.
Kerberos SPN	The Service Principal Name for the Kerberos Domain Controller.
LDAP Password	The password of the default LDAP user. It must be set if the LDAP server do not allow anonymous bind.
LDAP Port	The port for the LDAP server.
LDAP Server	The host name or IP address of the LDAP server.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Null To Unset	Use unset instead of NULL in CQL query when performing INSERT operations.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The maximum number of results to return per page from Cassandra Server.
Password	The password used to authenticate with Cassandra.
Port	The port for the Cassandra database.
Query Passthrough	This option passes the query to Cassandra as-is.

Readonly	You can use this property to enforce read-only access to Cassandra from the provider.
Row Scan Depth	The maximum number of rows to scan to look for the columns available in a table. Set this property to gain more control over how the provider applies data types to collections.
Search Base	The search base for your LDAPServer, used to look up users.
Search Filter	The search filter for looking up usernames in LDAP. The default setting is (uid=), When using Active Directory set the filter to (sAMAccountName=).
Server	The host name or IP address of the server hosting the Cassandra database.
SSL Client Cert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSL Client Cert Password	The password for the TLS/SSL client certificate.
SSL Client Cert Subject	The subject of the TLS/SSL client certificate.
SSL Client Cert Type	The type of key store containing the TLS/SSL client certificate.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Json Format	Whether to submit and return the JSON encoding for CQL data types.
User	The username used to authenticate with Cassandra.
Use SSL	This field sets whether SSL is enabled.
Varint To String	Map Cassandra VARINT to String value.

Allow Filtering

When true, slow-performing queries are processed on the server.

Data Type

bool

Default Value

false

Remarks

Cassandra by default does not allow filtering for queries that it predicts will have performance problems. These queries include filtering on a column that is not the primary key.

You can override the default behavior and rely on the server to process these queries by setting AllowFiltering to true.

Auth Scheme

The scheme used for authentication. Accepted entries are BASIC, DSE, KERBEROS, and LDAP.

Data Type

string

Default Value

"BASIC"

Remarks

Set this property to authenticate to open-source or DataStax Enterprise (DSE) Cassandra instances.

Together with Password and User, this field is used to authenticate against the server. BASIC is the default option. Use the following options to select your authentication scheme:

BASIC: Set this to authenticate with login credentials and Cassandra's built-in authentication.

DSE: Set this to authenticate with login credentials and the DSE Unified Authenticator.

KERBEROS: Set this to use Kerberos to authenticate.

LDAP: Set this to use LDAP to authenticate.

See the Getting Started section for guides to using each authentication method.

Case Sensitivity

Enable case sensitivity to the CQL sending to the server, if set to True, the identifiers in the CQL will be enclosed in double quotation marks.

Data Type

bool

Default Value

true

Remarks

By default, SQL is case-insensitive. However, Cassandra supports case-sensitive table and column names. Setting this property to True will enable you to retrieve tables and columns based on their case-sensitive names.

Consistency Level

The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success.

Data Type

string

Default Value

"ONE"

Remarks

The consistency level determines how many of the replicas of the data you are interacting with need to respond for the query to be considered a success. You need to specify the appropriate replicas in the Server property.

Below are the possible values:

ANY: At least one replica must return success in a write operation. This property guarantees that a write never fails; this consistency level delivers the lowest consistency and highest availability.

ALL: All replicas must respond. This property provides the highest consistency and the lowest availability.

ONE: At least one replica must respond. This is the default and suitable for most users, who do not typically require high consistency.

TWO: At least two replicas must respond.

THREE: At least three replicas must respond.

QUORUM: A quorum of nodes must respond. The QUORUM properties provide high consistency with some failure tolerance.

EACH_QUORUM: A quorum of nodes must respond where a quorum is calculated for each data center. This setting maintains consistency in each data center.

SERIAL: A quorum of replicas performs a consensus algorithm to allow lightweight transactions.

LOCAL_ONE: At least one replica in the local data center must respond.

LOCAL_SERIAL: The consensus algorithm is calculated for the local data center.

LOCAL_QUORUM: A quorum of nodes must respond where the quorum is calculated for the local data center.

Database

The name of the Cassandra keyspace.

Data Type

string

Default Value

""

Remarks

The name of the Cassandra keyspace containing the tables.

Default LDAP User

The default LDAP user used to connect to and communicate with the server, it must be set if the LDAP server do not allow anonymous bind.

Data Type

string

Default Value

""

Remarks

Specify the default LDAP user in case the LDAP server do not allow anonymous login.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Cassandra and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Flatten Arrays

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.

Data Type

string

Default Value

""

Remarks

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

Set FlattenArrays to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

For example, you can return an arbitrary number of elements from an array of strings:

```
["FLOW-MATIC","LISP","COBOL"]
```

When FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
languages_0	FLOW-MATIC

Flatten Objects

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Data Type

bool

Default Value

false

Remarks

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON. The property name is concatenated onto the object name with an underscore to generate the column name.

For example, you can flatten the nested objects below at connection time:

```
[
     { "grade": "A", "score": 2 },
     { "grade": "A", "score": 6 },
     { "grade": "A", "score": 10 },
     { "grade": "A", "score": 9 },
     { "grade": "B", "score": 14 }
]
```

When FlattenObjects is set to true and FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
grades_0_grade	A
grades_0_score	2

Kerberos KDC

The Kerberos Key Distribution Center (KDC) service used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos Key Distribution Center (KDC) service. The Kerberos Key Distribution Center (KDC) service is conventionally colocated with the domain controller. If Kerberos KDC is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using the config file krb5.conf, or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the configured domain name and host as a last resort.

Note: Windows authentication is supported in JRE 1.6 and above only.

The Kerberos properties are used when using Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos Key Distribution Center (KDC) service. The Kerberos Key Distribution Center (KDC) service is conventionally colocated with the domain controller. If Kerberos KDC is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using the config file krb5.conf, or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the configured domain name and host as a last resort.

Note: Windows authentication is supported in JRE 1.6 and above only.

Kerberos Realm

The Kerberos Realm used to authenticate the user with.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using SPNEGO or Windows Authentication. The Kerberos Realm is used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name. If Kerberos Realm is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using a config file (krb5.conf) or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the user-configured domain name and host as a last resort. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

The Kerberos properties are used when using SPNEGO or Windows Authentication. The Kerberos Realm is used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name. If Kerberos Realm is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using a config file (krb5.conf) or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the user-configured domain name and host as a last resort. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

Kerberos SPN

The Service Principal Name for the Kerberos Domain Controller.

Data Type

string

Default Value

""

Remarks

If the Service Principal Name on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, set the Service Principal Name here.

If the Service Principal Name on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, set the Service Principal Name here.

LDAP Password

The password of the default LDAP user. It must be set if the LDAP server do not allow anonymous bind.

Data Type

string

Default Value

""

Remarks

Specify the password of the default LDAP user.

LDAP Port

The port for the LDAP server.

Data Type

string

Default Value

"389"

Remarks

The port for the LDAP server.

LDAP Server

The host name or IP address of the LDAP server.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the LDAP server.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Null To Unset

Use unset instead of NULL in CQL query when performing INSERT operations.

Data Type

bool

Default Value

false

Remarks

In Cassandra 2.2 and above, when executing an INSERT query, a parameter value can be set to unset. Cassandra does not consider unset field values which helps to avoid tombstones.

When NULL values are inserted, it is possible to reach the tombstone threshold limits which causes an exception to be thrown when querying the data. Setting this property to true and submitting unset values avoids these tombstones from being created.

Note: This option is only available on INSERT operations as Cassandra does not support changing existing values to unset.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The maximum number of results to return per page from Cassandra Server.

Data Type

string

Default Value

"5000"

Remarks

The Pagesize property affects the maximum number of results to return per page from Cassandra when executing a query. A higher value will return more results per page, but may also cause more time consuming.

Password

The password used to authenticate with Cassandra.

Data Type

string

Default Value

""

Remarks

The password used to authenticate with Cassandra.

Port

The port for the Cassandra database.

Data Type

string

Default Value

"9042"

Remarks

The port for the Cassandra database.

Query Passthrough

This option passes the query to Cassandra as-is.

Data Type

bool

Default Value

false

Remarks

This option passes the query to Cassandra as-is.

Readonly

You can use this property to enforce read-only access to Cassandra from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Row Scan Depth

The maximum number of rows to scan to look for the columns available in a table. Set this property to gain more control over how the provider applies data types to collections.

Data Type

string

Default Value

"100"

Remarks

The columns in a table must be determined by scanning table rows. This value determines the maximum number of rows that will be scanned. The default value is 100.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

Search Base

The search base for your LDAPServer, used to look up users.

Data Type

string

Default Value

""

Remarks

The search base for your LDAPServer, used to look up users.

Search Filter

The search filter for looking up usernames in LDAP. The default setting is (uid=), When using Active Directory set the filter to (sAMAccountName=).

Data Type

string

Default Value

"uid="

Remarks

The search filter for looking up usernames in LDAP. The default setting is (uid=).

Server

The host name or IP address of the server hosting the Cassandra database.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the server hosting the Cassandra database. To connect to a distributed system, you can set Server to a comma-separated list of servers and ports, separated by colons. You will also need to set ConsistencyLevel.

Note that you must specify all of the servers required by your selected consistency level.

SSL Client Cert

The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The SSLClientCertType field specifies the type of the certificate store specified by SSLClientCert. If the store is password protected, specify the password in SSLClientCertPassword.

SSLClientCert is used in conjunction with the SSLClientCertSubject field in order to specify client certificates. If SSLClientCert has a value, and SSLClientCertSubject is set, a search for a certificate is initiated. Please refer to the SSLClientCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

SSL Client Cert Password

The password for the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

SSL Client Cert Subject

The subject of the TLS/SSL client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality

S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

SSL Client Cert Type

The type of key store containing the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.

PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Json Format

Whether to submit and return the JSON encoding for CQL data types.

Data Type

bool

Default Value

true

Remarks

Cassandra 2.2 introduced a CQL extension that allows you to JSON-encode CQL data types. By default, you use the JSON syntax to manipulate data and SELECT statements return JSON through the adapter. Set this property to false to use CQL literals to interact with Cassandra data.

The syntax for CQL literals has several differences from JSON. For example:

CQL strings are defined in single quotes, while JSON strings are defined in double quotes.

CQL sets, tuples, and lists are JSON-encoded as arrays.

User-defined types and CQL *uuid* types are JSON-encoded as objects.

Refer to the CQL documentation for more information on how to JSON-encode data types in your version of Cassandra. Below is an example SQL statement using JSON and CQL.

Format	Syntax
CQL	`INSERT INTO users (user_id, emails) VALUES(@user_id, @emails)`

	Parameters	
	user_id	frodo
	emails	{'f@baggins.com', 'baggins@gmail.com'}
JSON	`INSERT INTO users (user_id, emails) VALUES (@user_id, @emails)`	
	Parameters	
	user_id	frodo
	emails	["f@baggins.com", "baggins@gmail.com"])

Note that in queries to the adapter, you must use single quotes to define strings.

User

The username used to authenticate with Cassandra.

Data Type

string

Default Value

""

Remarks

The username used to authenticate with Cassandra.

Use SSL

This field sets whether SSL is enabled.

Data Type

bool

Default Value

false

Remarks

This field sets whether the adapter will attempt to negotiate TLS/SSL connections to the server. By default, the adapter checks the server's certificate against the system's trusted certificate store. To specify another certificate, set SSLServerCert.

Varint To String

Map Cassandra VARINT to String value.

Data Type

bool

Default Value

true

Remarks

Map Cassandra VARINT to String value.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Cassandra Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Accessing NoSQL Tables

The adapter implements Automatic Schema Discovery that is highly configurable. The following sections outline the adapter's defaults and link to ways to further customize.

Flattening Nested JSON

By default, the adapter projects columns over the properties of objects, including objects nested in objects. Arrays are returned as JSON strings, by default. You can use the following properties to access array elements, including objects nested in arrays.

FlattenArrays: Set this property to the number of array elements that you want to return as column values. You can also use this property with FlattenObjects to extract the properties of objects nested in arrays.

FlattenObjects: By default, this is true; that is, the properties of objects and nested objects are returned as columns. When you set FlattenArrays, objects nested in the specified array elements are also flattened and returned as columns.

Other mechanisms for accessing nested objects are detailed in NoSQL Database. See Advanced Settings for more control over data access at connection time.

Fine Tuning Data Access

You can use the following properties to gain greater control over Cassandra API features and the strategies the adapter uses to surface them:

AllowFiltering: Set this property to allow the server to process slow-performing searches.

UseJsonFormat: Set this property to use CQL literals instead of JSON.

QueryPassthrough: This property enables you to use native CQL statements instead of SQL.

RowScanDepth: This property determines the number of rows that will be scanned to detect column data types when generating table metadata. This property applies if you are working with the dynamic schemas generated from Automatic Schema Discovery or if you are using QueryPassthrough.

Connecting Through a Firewall or Proxy

To connect set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate set FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

NoSQL Database

Cassandra is a NoSQL database that provides high performance, availability, and scalability. However, these capabilities are not necessarily incompatible with a standards-compliant query language like SQL-92. The adapter models Cassandra tables into relational tables and translates SQL queries into calls to the Cassandra API, the CQL (Cassandra Query Language) binary protocol.

The equivalent of a table in Cassandra is a column family. Column families contain columns of related data. Like other NoSQL databases, Cassandra allows complex types of fields such as set, list, and map. A column family is a nested map data structure. This can be represented as a JSON object.

The adapter offers two ways to model Cassandra objects. The Automatic Schema Discovery scheme automatically finds the data types in a Cassandra object by scanning a configured number of rows of the object. You can use RowScanDepth, FlattenArrays, and FlattenObjects to control the relational representation of the tables in Cassandra.

Optionally, you can use Custom Schema Definitions to project your chosen relational structure on top of a Cassandra object. This allows you to define your chosen column names, their data types, and the location of their values in the Cassandra object.

Automatic Schema Discovery

The adapter automatically infers a relational schema by inspecting a Cassandra object. You can use the RowScanDepth property to define the number of rows the driver will scan to do so. The adapter models all top-level native properties of a collection as columns of an appropriate type.

For example, consider the following JSON representation of a Cassandra object that contains nested structures:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
  address: {street: "Main Street", city: "Chapel Hill", state:
"NC"},
  offices: ["Chapel Hill", "London", "New York"]
  annual_revenue: 35,600,000
}
```

The inferred types of the following JSON string are shown below.

Column Name	Data Type	Example Value
id	Integer	12
name	String	Lohia Manufacturers Inc.
address	String	{street: "Main Street", city: "Chapel Hill", state: "NC"}
offices	String	["Chapel Hill", "London", "New York"]
annual_revenue	Double	35,600,000

The columns identified during the discovery process also depend on the FlattenArrays and FlattenObjects properties. If FlattenObjects is set, all nested objects will be flattened into a series of columns. The preceding example will be represented by the following columns:

Column Name	Data Type	Example Value
id	Integer	12
name	String	Lohia Manufacturers Inc.
address_street	String	Main Street
address_city	String	Chapel Hill
address_state	String	NC
offices	String	["Chapel Hill", "London", "New York"]
annual_revenue	Double	35,600,000

The FlattenArrays property can be used to flatten array values into columns of their own. This is only recommended for arrays that are expected to be short, for example the coordinates below:

```
"coord": [ -73.856077, 40.848447 ]
```

The FlattenArrays property can be set to 2 to represent the array above as follows:

Column Name	Data Type	Example Value
coord_0	Float	-73.856077
coord_1	Float	40.848447

It is best to leave other unbounded arrays as they are and piece out the data for them as needed using JSON Functions.

JSON Functions

The adapter can return JSON structures as column values. The adapter enables you to use standard SQL functions to work with these JSON structures. The examples in this section use the following array:

```
[
     { "grade": "A", "score": 2 },
     { "grade": "A", "score": 6 },
     { "grade": "A", "score": 10 },
     { "grade": "A", "score": 9 },
     { "grade": "B", "score": 14 }
]
```

JSON_EXTRACT

The JSON_EXTRACT function can extract individual values from a JSON object. The following query returns the values shown below based on the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_EXTRACT(grades,'[0].grade') AS Grade,
JSON_EXTRACT(grades,'[0].score') AS Score FROM Students;
```

Column Name	Example Value
Grade	A
Score	2

JSON_COUNT

The JSON_COUNT function returns the number of elements in a JSON array within a JSON object. The following query returns the number of elements specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_COUNT(grades,'[x]') AS NumberOfGrades FROM
Students;
```

Column Name	Example Value
NumberOfGrades	5

JSON_SUM

The JSON_SUM function returns the sum of the numeric values of a JSON array within a JSON object. The following query returns the total of the values specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_SUM(score,'[x].score') AS TotalScore FROM
Students;
```

Column Name	Example Value
TotalScore	41

JSON_MIN

The JSON_MIN function returns the lowest numeric value of a JSON array within a JSON object. The following query returns the minimum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MIN(score,'[x].score') AS LowestScore FROM
Students;
```

Column Name	Example Value
LowestScore	2

JSON_MAX

The JSON_MAX function returns the highest numeric value of a JSON array within a JSON object. The following query returns the maximum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MAX(score,'[x].score') AS HighestScore FROM
Students;
```

Column Name	Example Value
HighestScore	14

JSON

The JSON function can be used to retrieve the entire table as a JSON string. See the following query and its result as an example:

```
SELECT JSON(*) from Customers;
```

The query above will return the entire table as shown.

```
{ "id": 12, "name": "Lohia Manufacturers Inc.", "address": {
"street": "Main Street", "city": "Chapel Hill", "state": "NC"},
"offices": [ "Chapel Hill", "London", "New York" ],
"annual_revenue": 35,600,000 }
```

Custom Schema Definitions

In addition to Automatic Schema Discovery the adapter also allows you to statically define the schema for your Cassandra object. Let's consider the schema below and extract out the nested properties as their own columns:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
  address: {street: "Main Street", city: "Chapel Hill", state:
"NC"},
  offices: ["Chapel Hill", "London", "New York"]
  annual_revenue: 35,600,000
}
{
  id: 15,
  name: "Piago Industries",
  address: {street: "Main Street", city: "Durham", state: "NC"},
  offices: ["Durham", "San Francisco"]
  annual_revenue: 42,600,000
}
```

Custom Schema Definition

You can define a custom schema to extract out nested properties as their own columns. The following schema uses the other:periodpath property to define where the data for a particular column should be retrieved from. Using this model you can flatten arbitrary levels of hierarchy.

The table attribute specifies the table to parse. The table attribute gives you the flexibility to use multiple schemas for the same table. If table is not specified, the filename determines the table that is parsed.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">
<rsb:info title="Customers" description="Customers">
  <!-- Column definitions -->
 <attr name="Id"        xs:type="int" key="true"
/>
 <attr name="Name"       xs:type="string"
/>
  <attr name="Street"          xs:type="string"
other:periodpath="address.street" />
  <attr name="City"            xs:type="string"
other:periodpath="address.city"   />
```

```
  <attr name="PrimaryOffice"   xs:type="string"
other:periodpath="offices.0"      />
  <attr name="SecondaryOffice" xs:type="string"
other:periodpath="offices.1"      />
</rsb:info>

<rsb:set attr="table" value="customers"/>

</rsb:script>
```

In Custom Schema Example, you will find the complete schema that contains the example above.

Custom Schema Example

In this section is a complete schema. The info section enables a relational view of a Cassandra object. For more details, see Custom Schema Definitions. The table below allows the SELECT command as implemented in the GET section of the schema below. The operations, such as cassandraadoExecuteSelect, are internal implementations.

Use the table attribute to specify the name of the table you want to parse. You can use the table attribute to define multiple schemas for the same table.

If table is not specified, the filename determines the table that is parsed.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

  <rsb:info title="Customers" description="Customers">
    <!-- Column definitions -->
  <attr name="Id"       xs:type="int" key="true"
/>
   <attr name="Name"       xs:type="string"
/>
    <attr name="Street"          xs:type="string"
other:periodpath="address.street" />
    <attr name="City"            xs:type="string"
other:periodpath="address.city"   />
    <attr name="PrimaryOffice"   xs:type="string"
other:periodpath="offices.0"      />
    <attr name="SecondaryOffice" xs:type="string"
other:periodpath="offices.1"      />
  </rsb:info>
  <rsb:set attr="table" value="customers"/>
  <rsb:script method="GET">
    <rsb:push op="cassandraadoExecuteSelect"/>
  </rsb:script>

</rsb:script>
```

System Tables

Query the following system tables to access schema information, information on data source functionality, and batch operation statistics.

Schema Tables

The following tables return database metadata for Cassandra:

sys_catalogs: Lists the available databases.

sys_schemas: Lists the available schemas.

sys_tables: Lists the available tables.

sys_tablecolumns: Describes the columns of the available tables.

sys_views: Lists the available views.

sys_viewcolumns: Describes the columns of available views.

sys_procedures: Describes the available stored procedures.

sys_procedureparameters: Describes stored procedure parameters.

sys_keycolumns: Describes the primary and foreign keys.

sys_indexes: Describes the available indexes.

Data Source Tables

The following tables return information about how to connect to and query the data source:

sys_connection_props: Returns information on the available connection properties.

sys_sqlinfo: Describes the SELECT queries that the adapter can offload to the data source.

Query Information Tables

Query statistics are available for data modification queries, including batch operations.

sys_identity: Returns information about batch operations or single updates.

sys_catalogs

Lists the available databases.

The following query retrieves all databases determined by the connection string:

```
SELECT * FROM sys_catalogs
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.

sys_schemas

Lists the available schemas.

```
SELECT * FROM sys_schemas
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.
SchemaName	*String*	The schema name.

sys_tables

Lists the available tables.

```
SELECT * FROM sys_tables
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table.
TableType	*String*	The table type.
Description	*String*	A description of the table.

sys_tablecolumns

Describes the columns of the available tables.

The following query returns the columns and data types for the Account table:

```
SELECT ColumnName, DataTypeName FROM sys_tablecolumns WHERE
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table containing the column.
ColumnName	*String*	The column name.
DataTypeName	*String*	The data type name.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	A brief description of the column.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.

IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_views

Lists the available views.

```
SELECT TableName FROM sys_views
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
TableType	*String*	The type of the view.
Description	*String*	A description of the view.

sys_viewcolumns

Describes the columns of the available views.

The following query returns the columns and data types for a specified view:

```
SELECT ColumnName, DataTypeName FROM sys_viewolumns WHERE
TableName='MyView'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
ColumnName	*String*	The name of the column.
DataTypeName	*String*	The name of the data type.

DataType	*Int32*	An integer indicating the data type of the column. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	The column description.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_procedures

Lists the available stored procedures.

```
SELECT * FROM sys_procedures
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the stored procedure.
SchemaName	*String*	The schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure.

Description	*String*	A description of the stored procedure.

sys_procedureparameters

Describes stored procedure parameters.

The following query returns information about all of the input parameters for the SelectEntries stored procedure:

```
SELECT * FROM sys_procedureparameters WHERE
ProcedureName='SelectEntries' AND Direction=1 OR Direction=2
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the stored procedure.
SchemaName	*String*	The name of the schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure containing the parameter.
ColumnName	*String*	The name of the stored procedure parameter.
Direction	*Int32*	An integer corresponding to the type of the parameter: input (1), input/output (2), or output(4). input/output type parameters can be both input and output parameters.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The number of characters allowed for character data. The number of digits allowed for numeric data.
NumericPrecision	*Int32*	The maximum precision for numeric data. The column length in characters for character and date-time data.
NumbericScale	*Int32*	The number of digits to the right of the decimal point in numeric data.

IsNullable	*Boolean*	Whether the parameter can contain null.
Description	*String*	The description of the parameter.
Ordinal	*Int32*	The index of the parameter.

sys_keycolumns

Describes the primary and foreign keys.

The following query retrieves the primary key for the Account table:

```
SELECT * FROM sys_keycolumns WHERE IsKey='True' AND
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the key.
SchemaName	*String*	The name of the schema containing the key.
TableName	*String*	The name of the table containing the key.
ColumnName	*String*	The name of the key column.
IsKey	*Boolean*	Whether the column is a primary key in the table referenced in the TableName table.
IsForeignKey	*Boolean*	Whether the column is a foreign key referenced in the TableName table.
ReferencedCatalogName	*String*	The database containing the primary key.
ReferencedSchemaName	*String*	The schema containing the primary key.
ReferencedTableName	*String*	The table containing the primary key.
ReferencedColumnName	*String*	The column name of the primary key.

sys_indexes

Describes the available indexes. By filtering on indexes, you can write more selective queries with faster query response times.

The following query retrieves all indexes that are not primary keys:

```
SELECT * FROM sys_indexes WHERE IsPrimary='false'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the index.
SchemaName	*String*	The name of the schema containing the index.
TableName	*String*	The name of the table containing the index.
IndexName	*String*	The index name.
ColumnName	*String*	The name of the column associated with the index.
IsUnique	*Boolean*	True if the index is unique. False otherwise.
IsPrimary	*Boolean*	True if the index is a primary key. False otherwise.
Type	*Int16*	An integer value corresponding to the index type: statistic (0), clustered (1), hashed (2), or other (3).
SortOrder	*String*	The sort order: A for ascending or D for descending.
OrdinalPosition	*Int16*	The sequence number of the column in the index.

sys_connection_props

Returns information on the available connection properties and those set in the connection string.

The following query retrieves all connection properties that have been set in the connection string or set through a default value:

```
SELECT * FROM sys_connection_props WHERE Value <> ''
```

Columns

Name	Type	Description

Name	*String*	The name of the connection property.
ShortDescription	*String*	A brief description.
Type	*String*	The data type of the connection property.
Default	*String*	The default value if one is not explicitly set.
Values	*String*	A comma-separated list of possible values. A validation error is thrown if another value is specified.
Value	*String*	The value you set or a preconfigured default.
Required	*Boolean*	Whether the property is required to connect.
Category	*String*	The category of the connection property.
IsSessionProperty	*String*	Whether the property is a session property, used to save information about the current connection.

sys_sqlinfo

Describes the SELECT query processing that the adapter can offload to the data source.

Collaborative Query Processing

When working with data sources that do not support SQL-92, you can query the sys_sqlinfo view to determine the query capabilities of the underlying APIs, expressed in SQL syntax. See SQL Compliance for SQL syntax details.

Discovering the Data Source's SELECT Capabilities

Below is an example data set of SQL capabilities. Some aspects of SELECT functionality are returned in a comma-separated list if supported; otherwise, the column contains NO.

Name	**Possible Values**
AGGREGATE_FUNCTIONS	AVG, COUNT, MAX, MIN, SUM, DISTINCT

COUNT	YES
SUPPORTED_OPERATORS	=, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR
GROUP_BY	NO_RELATION
OUTER_JOINS	YES
OJ_CAPABILITIES	NESTED, LEFT, RIGHT, INNER, NOT_ORDERED, ALL_COMPARISON_OPS
SUBQUERIES	COMPARISON, EXISTS, IN, CORRELATED_SUBQUERIES, QUANTIFIED
STRING_FUNCTIONS	ASCII,CHAR,CONCAT,LEFT,LTRIM,REPLACE,RIGHT,RTRIM,SOUNDEX,SPACE,SUBSTRING
NUMERIC_FUNCTIONS	ABS,ACOS,ASIN,ATAN,CEILING,COS,COT,DEGREES,EXP,FLOOR,LOG,LOG10,PI,POWER,RADIANS,RAND,ROUND,SIGN,SIN,SQRT,TAN
TIMEDATE_FUNCTIONS	CURRENT_DATE,CURRENT_TIMESTAMP,MONTH,YEAR
IDENTIFIER_QUOTE_OPEN_CHAR	[
IDENTIFIER_QUOTE_CLOSE_CHAR	]

The following query retrieves the operators that can be used in the WHERE clause:

```
SELECT * FROM sys_sqlinfo WHERE Name='SUPPORTED_OPERATORS'
```

Note that individual tables may have different limitations or requirements on the WHERE clause; refer to the NoSQL Database section for more information.

Columns

Name	Type	Description
NAME	*String*	A component of SQL syntax, or a capability that can be processed on the server.
VALUE	*String*	Detail on the supported SQL or SQL syntax.

sys_identity

Returns information about attempted modifications. The following query retrieves the Ids of the modified rows in a batch operation.

```
SELECT * FROM sys_identity
```

Columns

Name	Type	Description
Id	*String*	The database-generated Id returned from a data modification operation.
Batch	*String*	An identifier for the batch. 1 for a single operation.
Operation	*String*	The result of the operation in the batch: INSERTED, UPDATED, or DELETED.
Message	*String*	SUCCESS or an error message if the update in the batch failed.

SQL Compliance

The Cassandra Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See NoSQL Database for information on the capabilities of the Cassandra API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Cassandra adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
```

```
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | IN } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Account
```

Rename a column:

```
SELECT "Name" AS MY_Name FROM Account
```

Search data:

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

The Cassandra APIs support the following operators in the WHERE clause: =, >, <, >=, <=, IN.

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM Account
```

Summarize data:

```
SELECT  MAX(AnnualRevenue) FROM Account
```

See Aggregate Functions for details.

Sort a result set in ascending order:

```
SELECT Id, Name FROM Account  ORDER BY Name ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM Account WHERE Industry = 'Floppy Disks'
```

AVG

Returns the average of the column values.

```
SELECT  AVG(AnnualRevenue) FROM Account WHERE Industry = 'Floppy
Disks'
```

MIN

Returns the minimum column value.

```
SELECT MIN(AnnualRevenue) FROM Account WHERE Industry = 'Floppy
Disks'
```

MAX

Returns the maximum column value.

```
SELECT  MAX(AnnualRevenue) FROM Account WHERE Industry = 'Floppy
Disks'
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(AnnualRevenue) FROM Account WHERE Industry = 'Floppy
Disks'
```

Projection Functions

JSON_AVG(json, jsonpath)

Computes the average value of a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_COUNT(json, jsonpath)

Returns the number of elements in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MAX(json, jsonpath)

Gets the maximum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MIN(json, jsonpath)

Gets the minimum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_SUM(json, jsonpath)

Computes the sum of the elements in a JSON within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_EXTRACT(json, jsonpath)

Selects any value in a JSON array or object. The path to the array is specified in the jsonpath argument. Return value is numeric or null.

json: The JSON document to extract.

jsonpath: The XPath used to select the nodes. The JSONPath must be a string constant. The values of the nodes selected will be returned in a token-separated list.

XML_EXTRACT(xml, xpath [, separator])

Extracts an XML document using the specified XPath to flatten the XML. A comma is used to separate the outputs by default, but this can be changed by specifying the third parameter.

xml: The XML document to extract.

xpath: The XPath used to select the nodes. The nodes selected will be returned in a token-separated list.

separator: The optional token used to separate the items in the flattened response. If this is not specified, the separator will be a comma.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Name INTO "csv://Account.txt" FROM "Account" WHERE
Industry = 'Floppy Disks'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Name INTO "csv://Account.txt;delimiter=tab" FROM
"Account" WHERE Industry = 'Floppy Disks'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Account SET Name='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:cassandra:Database=MyCassandraDB
;Port=9042;Server=127.0.0.1;",);
String cmd = "DELETE FROM Account WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

TDV Microsoft Dynamics CRM Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to Microsoft Dynamics CRM

To connect, set Url to the root URL of your organization and set User, Password, and CRMVersion.

Internet Facing Deployments

For Dynamics CRM with IFD, set InternetFacingDeployment to true.

Authenticating to CRM On-Premises

Additionally for CRM on-premises, select an authentication method. By default, the adapter uses Windows (NTLM) authentication. To use another authentication type, such as Kerberos delegation, set AuthScheme.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Scheme	The authentication scheme used. Accepted entries are NTLM, BASIC, DIGEST, NONE, and KERBEROSDELEGATION.
Caller Id	The Id of a user to impersonate when inserting or updating new records.

CRM Version	The type of Dynamics CRM server to which you are connecting. Accepted entries are CRM2011+, CRMOnline.
Default Precision	The currency precision that is used for pricing throughout the system. Valid values are 0-4 and Auto. If the value is Auto, the default value will be retrieved from the Dynamics CRM server.
Device Credential Location	The path to a file where device credentials are stored. If the file does not exist or is empty a new set of credentials will be generated and stored in that file.
Device Credential Password	The password used to encrypt or decrypt the credentials stored in the credentials file.
Device Name	The device name used to authenticate along with a Windows Live Id.
Device Password	The device password used to authenticate along with a Windows Live Id.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Include Calculated Columns	This option controls whether the driver returns the Calculated Columns defined on a table. Only applicable for CRM 2015+.
Internet Facing Deployment	Whether you are connecting to an Internet Facing Deployment (IFD) for CRM.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Organization Name	The name of the organization. In Dynamics CRM 4.0 without IFD, the organization is specified in the URL; for example, http://website/organizationname. In Dynamics CRM 4.0 with IFD, this property must be set. In other versions of CRM, this property is optional.

Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The number of rows to return per page from the data source.
Password	The password used to authenticate to the Dynamics CRM site.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Query Method	The method to use when querying data from Dynamics CRM. In most cases FetchXML will work with all tables.
Query Passthrough	This option passes the Fetch XML query to Dynamics CRM as-is.
Readonly	You can use this property to enforce read-only access to Dynamics CRM from the provider.
SSL Cert	The certificate used for client authentication.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
STSURL	The URL of the security token service (STS). This value is detected automatically.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Url	The root URL of the organization. For example, a CRM 4.0 or CRM 2011 URL will resemble http://MySite/MyOrganization. For CRM Online, the URL will resemble https://myOrg.crm.dynamics.com/.
Use Display Names	Boolean determining if the display names for the columns should be used instead of the API names.
Use Name For Picklist Value	Boolean determining if the string value should be used for picklist field values instead of integers.
User	The user who is authenticating to the Dynamics CRM site.
Use Simple Names	Boolean determining if simple names should be used for tables and columns.

Auth Scheme

The authentication scheme used. Accepted entries are NTLM, BASIC, DIGEST, NONE, and KERBEROSDELEGATION.

Data Type

string

Default Value

"NTLM"

Remarks

Together with Password and User, this field is used to authenticate against an on-premises Dynamics CRM 4.0 server. This property will not be used for other versions of CRM. NTLM is the default option. Use the following options to select your authentication scheme:

NTLM: Set this to use your Windows credentials for authentication.

BASIC: Set this to use HTTP Basic authentication.

DIGEST: Set this to use HTTP Digest authentication.

KERBEROSDELEGATION: Set this to use delegation through the Kerberos protocol. Set the

User and Password of the account you want to impersonate.

Caller Id

The Id of a user to impersonate when inserting or updating new records.

Data Type

string

Default Value

""

Remarks

The Id of a user to impersonate when inserting or updating new records. All records modified with this set appear as if they were edited by the impersonated user.

CRM Version

The type of Dynamics CRM server to which you are connecting. Accepted entries are CRM2011+, CRMOnline.

Data Type

string

Default Value

"CRM2011+"

Remarks

The type of Dynamics CRM server to which you are connecting. Accepted entries are CRM2011+ or CRMOnline. A value of CRMOnline is required to connect using the Office 365 STS.

Set InternetFacingDeployment to connect to an IFD instance of CRM.

Default Precision

The currency precision that is used for pricing throughout the system. Valid values are 0-4 and Auto. If the value is Auto, the default value will be retrieved from the Dynamics CRM server.

Data Type

string

Default Value

"2"

Remarks

The decimal precision that is used for currencies throughout the system. Valid values are 0-4 and Auto. If the value is Auto, the default value will be retrieved from the Microsoft Dynamics CRM server.

Device Credential Location

The path to a file where device credentials are stored. If the file does not exist or is empty a new set of credentials will be generated and stored in that file.

Data Type

string

Default Value

""

Remarks

Dynamics CRM uses claims-based authentication when authenticating using a Windows Live Id. This requires device credentials, which consist of a DeviceName and a DevicePassword. The device credentials need to be generated only once and can be reused. It is customary for each application to have its own set of device credentials.

The credentials in the DeviceCredentialLocation may be protected using the DeviceCredentialPassword. If the DeviceCredentialPassword is specified, then it will be used to encrypt and decrypt the device credentials. If it is not specified, then the credentials are stored in plaintext.

If the DeviceCredentialLocation is not specified and neither are DeviceName and DevicePassword, then the adapter will generate the credentials for onetime use. This can be inefficient for repeated use.

Device Credential Password

The password used to encrypt or decrypt the credentials stored in the credentials file.

Data Type

string

Default Value

""

Remarks

Dynamics CRM uses claims-based authentication when authenticating using a Windows Live Id. This requires device credentials, which consist of a DeviceName and a DevicePassword. The device credentials need to be generated only once and can be reused. It is customary for each application to have its own set of device credentials.

The credentials in the DeviceCredentialLocation may be protected using the DeviceCredentialPassword. If the DeviceCredentialPassword is specified, then it will be used to encrypt and decrypt the device credentials. If it is not specified, then the credentials are stored in plaintext.

Device Name

The device name used to authenticate along with a Windows Live Id.

Data Type

string

Default Value

""

Remarks

Dynamics CRM uses claims-based authentication when authenticating using a Windows Live Id. This requires device credentials, which consist of a DeviceName and a DevicePassword. The device credentials need to be generated only once and can be reused. It is customary for each application to have its own set of device credentials.

You may specify device credentials using the DeviceName and DevicePassword properties or using DeviceCredentialLocation. If DeviceName and DevicePassword are specified, the contents of the DeviceCredentialLocation are ignored.

Device Password

The device password used to authenticate along with a Windows Live Id.

Data Type

string

Default Value

""

Remarks

Dynamics CRM uses claims-based authentication when authenticating using a Windows Live Id. This requires device credentials, which consist of a DeviceName and a DevicePassword. The device credentials need to be generated only once and can be reused. It is customary for each application to have its own set of device credentials.

You may specify device credentials using the DeviceName and DevicePassword properties or using DeviceCredentialLocation. If DeviceName and DevicePassword are specified, the contents of the DeviceCredentialLocation are ignored.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Microsoft Dynamics CRM and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Include Calculated Columns

This option controls whether the driver returns the Calculated Columns defined on a table. Only applicable for CRM 2015+.

Data Type

bool

Default Value

true

Remarks

This option controls whether the driver returns the Calculated Columns defined on a table. Only applicable for CRM 2015+.

Internet Facing Deployment

Whether you are connecting to an Internet Facing Deployment (IFD) for CRM.

Data Type

bool

Default Value

false

Remarks

Set this to true if you are connecting to an Internet Facing Deployment (IFD) for CRM.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Organization Name

The name of the organization. In Dynamics CRM 4.0 without IFD, the organization is specified in the URL; for example, http://website/organizationname. In Dynamics CRM 4.0 with IFD, this property must be set. In other versions of CRM, this property is optional.

Data Type

string

Default Value

""

Remarks

The following table is a description for setting this property in each Dynamics CRM version.

Dynamics CRM 4.0 without IFD	Optional. If this property is not set, the organization name can be retrieved from the URL.
CRM 4.0 with IFD	Required. This property must be set.
Dynamics CRM 2011 on-premises	Optional. If this property is not set, the organization name can be retrieved from the URL.
Dynamics CRM 2011 with IFD	Optional.
Dynamics CRM 2011 without IFD	Optional.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The number of rows to return per page from the data source.

Data Type

string

Default Value

"500"

Remarks

The number of rows to return per page from the data source.

Password

The password used to authenticate to the Dynamics CRM site.

Data Type

string

Default Value

""

Remarks

The password used to authenticate to the Dynamics CRM site.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Query Method

The method to use when querying data from Dynamics CRM. In most cases FetchXML will work with all tables.

Data Type

string

Default Value

"FetchXML"

Remarks

The method to use when querying data from Dynamics CRM. In most cases FetchXML will work with all tables. However, QueryExpression may be specified as an alternative.

Query Passthrough

This option passes the Fetch XML query to Dynamics CRM as-is.

Data Type

bool

Default Value

false

Remarks

Set this option to write your own Fetch XML queries instead of SQL statements.

Readonly

You can use this property to enforce read-only access to Dynamics CRM from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Cert

The certificate used for client authentication.

Data Type

string

Default Value

""

Remarks

If using an SSL connection, this property can be used to specify the SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected. This can take the form of a full PEM certificate, the path to a file containing the certificate, the public key, the MD5 thumbprint, or the SHA1 thumbprint. If not specified, any valid certificate will be accepted. Use '*' to signify to accept all certificates.

SSLCert is used for specifying a private key certificate for HTTP client authentication.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

STSURL

The URL of the security token service (STS). This value is detected automatically.

Data Type

string

Default Value

""

Remarks

The adapter will automatically select the correct STSURL. However, if you need to conserve network resources, you can define this value in STSURL to avoid retrieving the URL on every connection. To retrieve this value, call the GetSTSUrl stored procedure.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Url

The root URL of the organization. For example, a CRM 4.0 or CRM 2011 URL will resemble http://MySite/MyOrganization. For CRM Online, the URL will resemble https://myOrg.crm.dynamics.com/.

Data Type

string

Default Value

""

Remarks

The root URL of the organization. For example, a CRM 4.0 or CRM 2011 URL will resemble http://MySite/MyOrganization. For CRM Online, the URL will resemble https://myOrg.crm.dynamics.com/.

Use Display Names

Boolean determining if the display names for the columns should be used instead of the API names.

Data Type

bool

Default Value

false

Remarks

Boolean determining if the display names for the columns should be used instead of the API names.

Use Name For Picklist Value

Boolean determining if the string value should be used for picklist field values instead of integers.

Data Type

bool

Default Value

true

Remarks

Boolean determining if the string value should be used for picklist field values instead of integers.

User

The user who is authenticating to the Dynamics CRM site.

Data Type

string

Default Value

""

Remarks

The user who is authenticating to the Dynamics CRM site.

Use Simple Names

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

If true, tables and columns returned will be renamed to make them easier to read and to specify in a query. Requirements for using [] or '' are removed. If false, the tables and columns will appear as they do in Dynamics CRM.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Microsoft Dynamics CRM Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Fine Tuning Data Access

Use the following connection properties to control column name identifiers and other aspects of data access useful in more advanced integrations:

IncludeCalculatedColumns

UseNameForPicklistValue

UseDisplayNames

UseSimpleNames

DefaultPrecision

CallerId

Securing Microsoft Dynamics CRM Connections

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

By default, the adapter will connect through the Windows system proxy if one is defined. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Microsoft Dynamics CRM Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Microsoft Dynamics CRM API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

UPSERT Statements

An UPSERT updates a record if it exists and inserts the record if it does not. See UPSERT Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing. Dynamics 365 and on-premise instances since CRM 2013 support bulk operations. The adapter abstracts the Microsoft Dynamics CRM bulk API into SQL. The following sections describe the SQL you can use to execute bulk operations to Microsoft Dynamics CRM.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Microsoft Dynamics CRM adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
```

```
  } [ , ... ]
  [ [
      INNER | { { LEFT | RIGHT | FULL  } [ OUTER ] }
    ] JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
    [
      { OFFSET | , }
      <expression>
    ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | <> | < | > | <= | >= | IS NULL | IS
NOT NULL | CONTAINS | LIKE | NOT LIKE | IN | NOT IN | AND | OR } [
<expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Lead
```

Rename a column:

```
SELECT "FirstName" AS MY_FirstName FROM Lead
```

Search data:

```
SELECT * FROM Lead WHERE FirstName <> 'Bob';
```

The Microsoft Dynamics CRM APIs support the following operators in the WHERE clause: =, !=, <>, <, >, <=, >=, IS NULL, IS NOT NULL, CONTAINS, LIKE, NOT LIKE, IN, NOT IN, AND, OR.

```
SELECT * FROM Lead WHERE FirstName <> 'Bob';
```

Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM Lead
```

Return the number of unique items matching the query criteria:
```
SELECT COUNT(DISTINCT FirstName) FROM Lead
```

Summarize data:
```
SELECT FirstName, MAX(Revenue) FROM Lead  GROUP BY FirstName
```

See Aggregate Functions for details.

Retrieve data from multiple tables.
```
SELECT Account.Id, Account.Name, Equipment.Id AS Eid,
Equipment.Name AS Ename FROM Account JOIN Equipment ON Equipment.Id
= Account.PreferredEquipmentId_Id
```

See JOIN Queries for details.

Sort a result set in ascending order:
```
SELECT Id, FirstName FROM Lead  ORDER BY FirstName ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.
```
SELECT COUNT(*) FROM Lead WHERE FirstName <> 'Bob'
```

COUNT_DISTINCT

Returns the number of distinct, non-null field values matching the query criteria.
```
SELECT COUNT_DISTINCT(Id) AS DistinctValues FROM Lead WHERE
FirstName <> 'Bob'
```

AVG

Returns the average of the column values.
```
SELECT FirstName, AVG(Revenue) FROM Lead WHERE FirstName <> 'Bob'
GROUP BY FirstName
```

MIN

Returns the minimum column value.

```
SELECT MIN(Revenue), FirstName FROM Lead WHERE FirstName <> 'Bob'
GROUP BY FirstName
```

MAX

Returns the maximum column value.

```
SELECT FirstName, MAX(Revenue) FROM Lead WHERE FirstName <> 'Bob'
GROUP BY FirstName
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(Revenue) FROM Lead WHERE FirstName = 'Bob'
```

JOIN Queries

The adapter supports JOIN queries based on Dynamics CRM relationships. JOIN queries in Dynamics CRM can only be executed against related entities.

Dynamics CRM entities can be linked using relationships. The standard Dynamics CRM entities already have relationships defined for them. You can define relationships for your custom entities. The adapter supports standard SQL syntax instead of proprietary FetchXML to allow easy integration with a wide variety of SQL tools.

Inner Joins

Inner joins are the default join when the JOIN keyword is specified. The INNER and NATURAL keywords are also supported. The following query returns the Names of all Accounts that have Contacts and the FirstNames of those Contacts.

```
SELECT Account.Id, Account.Name, Contact.FirstName,
Contact.LastName FROM Account JOIN Contact ON Account.Id =
Contact.AccountId_Id
```

Left Join

Left joins can be executed with the LEFT JOIN and LEFT OUTER JOIN keywords. The following returns all Accounts and the Equipment Id for any preferred Equipment defined for that Account:

```
SELECT Account.Id, Account.Name, Equipment.Id AS Eid,
Equipment.Name AS Ename FROM Account LEFT JOIN Equipment ON
Account.PreferredEquipmentid_id = Equipment.Id WHERE Account.Name
= 'Adventure Works (sample)'
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, FirstName INTO "csv://Lead.txt" FROM "Lead" WHERE
FirstName = 'Bob'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, FirstName INTO "csv://Lead.txt;delimiter=tab" FROM
"Lead" WHERE FirstName = 'Bob'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Lead (FirstName) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
```

```
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Lead SET FirstName='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1045625d-99ee-e011-a272-00155d01ad6b");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

UPSERT Statements

An UPSERT statement will update an existing record or create a new record if an existing record is not identified.

UPSERT Syntax

The UPSERT syntax is the same as for insert. Microsoft Dynamics CRM uses the input provided in the VALUES clause to determine whether the record already exists. If the record does not exist, all columns required to insert the record must be specified. See Data Model for any table-specific information.

```
UPSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "UPSERT INTO Lead (FirstName) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:dynamicscrm:User=myuseraccount;P
assword=mypassword;URL=https://myOrg.crm.dynamics.com/;CRM
Version=CRM Online;",);
String cmd = "DELETE FROM Lead WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1045625d-99ee-e011-a272-00155d01ad6b");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The Microsoft Dynamics CRM Adapter models Microsoft Dynamics CRM entities in relational tables and stored procedures. The table definitions are dynamically obtained based on your Dynamics CRM organization settings.

A full list is available upon customer request.

TDV Microsoft Dynamics GP Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to Microsoft Dynamics GP

To authenticate set User and Password.

Connecting to Microsoft Dynamics GP

To connect set the Url to the Web services endpoint; for example, http://{servername}:{port}/Dynamics/GPService. Additionally, set CompanyId; you can obtain this value in the company setup window: Click Tools -> Setup -> Company.

Fine-Tuning Data Access

The adapter returns data summaries by default to save performance. Set LookupIds to true to return details such as line items; however, note that entities must be retrieved one at a time.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Company Id	The unique identifier of the company to access as a data source.

Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Ignore Lookup Id Errors	A boolean indicating if errors on Ids that are looked up should be ignored.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Lookup Ids	A boolean indicating if ids should be looked up.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password for the user connecting to Dynamics GP.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.

Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Url	The URL of the Dynamics GP server.
User	The user that is authenticating to the Dynamics GP Web Services.

Company Id

The unique identifier of the company to access as a data source.

Data Type

string

Default Value

""

Remarks

This property is required to connect. You can obtain this value in the company setup window: Click Tools -> Setup -> Company.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Microsoft Dynamics GP and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Ignore Lookup Id Errors

A boolean indicating if errors on Ids that are looked up should be ignored.

Data Type

bool

Default Value

true

Remarks

Sometimes Microsoft Dynamics GP will throw exceptions when looking up a specific Id. This can indicate that there is an internal problem with the record, even though it is listed just fine along with all of the rest when it is not looked up by Id. To allow the Microsoft Dynamics GP Adapter to work without interruption, these errors are ignored by default and simply logged. Set this property to false to force the Microsoft Dynamics GP Adapter to throw an error when there is a problem.

This property is also used when retrieving data from a child table. Child tables must always be retrieved by id as they do not come back in the summary responses that list all parent items.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Lookup Ids

A boolean indicating if ids should be looked up.

Data Type

bool

Default Value

false

Remarks

Most entities are retrieved via a summary from DynamicsGP. However, these summaries are incomplete. To get back more details such as line items, entities must be retrieved one at a time by Id. Setting this value to true will cause the tool to perform this lookup automatically, although it will harm performance.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password for the user connecting to Dynamics GP.

Data Type

string

Default Value

""

Remarks

This property is required to connect.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Url

The URL of the Dynamics GP server.

Data Type

string

Default Value

""

Remarks

This property is required to connect. The full path must be specified. For example, http://{servername}:{port}/Dynamics/GPService.

User

The user that is authenticating to the Dynamics GP Web Services.

Data Type

string

Default Value

""

Remarks

This property is required to connect.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Microsoft Dynamics GP Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Changes in the 2016 Version

The Microsoft Dynamics GP Adapter 2017 verion has been rewritten to automatically support the available entites in Microsoft Dynamics GP based on the metadata detected. This means far more tables are available, but there are a number of changes to be aware of.

Enhanced SQL Support

The Microsoft Dynamics GP Adapter 2017 version fundamentally changes how queries are typically executed. In the past, queries were limited to only what the Microsoft Dynamics GP SDK supports. Now by default the SupportEnhancedSQL connection property will be set to true. This will cause any more advanced filters than what Microsoft Dynamics GP can directly handle to be handled client side, after results are returned.

Table/Column Definitions

The benefit of automatically reading all tables and columns from Microsoft Dynamics GP is that any tables or columns that were not formerly available will be available. The drawback is that this makes it impossible to maintain the consistency of the formerly hardcoded, tables and columns in the previous version. If you are upgrading, assume all tables and columns have been affected and will not have the same definitions as the previous version.

LookupIds

In Microsoft Dynamics GP, many of the entities can only be requested more than one at a time using "summary" requests. These summary requests will include only a small amount of data associated with the entity. To retrieve all data, the entites must be retrieved by Id. Examples of entities affected by this are CreditMemos, PurchaseInvoices, PurchaseOrders, etc. In the previous version, these limitations were pushed to the user to handle. In the 2017 version, they are automatically handled via LookupIds.

The LookupIds connection property can be used to automatically retrieve individual entities by Id when it is detected that a summary request would need to be made and only a partial result will come back. The benefit is that all columns requested that have data will come back with data. The drawback to this is that it will cause far more requests than simply the summary request, which will noticably impact performance. If only basic fields that come back with a summary request are required, then LookupIds can be set to false.

Child Tables

Child tables refer to tables that are not complete entities on their own. For instance, CustomerAddresses, PurchaseInvoiceLines, and SalesDocumentTaxes are all examples of child tables. The data retrieved for child tables actually comes from the parent. The parent is the prefixed table name. So for instance, a SalesDocument is the parent for SalesDocumentTaxes. There is no individual request that will give back just the data for a child table. They are essentially just collections of data that come back from the parent.

In order to make working with these child collections easier, we have exposed them in these child tables. Child tables always require a request by Id from Microsoft Dynamics GP as these collections are not available from the summary responses. In the previous version, child tables always required the user to specify the parent id to retrieve data. In 2017, child tables do not have this restriction. Regardless of the value for LookupIds, child tables will always perform a lookup by id. This makes requests for data equally as expensive as retrieving data from the parent with LookupIds set to true.

Paging

Microsoft Dynamics GP does not contain a mechanism for performing true paging. Instead, it simply returns the first 1000 results. Beyond that, paging can only be accomplished by specifying criteria and submitting multiple requests.

In the previous version, paging was handled by using the last returned primary key and requesting only values with a greater primary key in subsequent pages. However, Microsoft Dynamics GP makes no guarantees as to the order of data and this could (in some cases) cause issues if lower primary keys came at the end of a given page. To avoid this issue, the Microsoft Dynamics GP Adapter has been updated to use the highest primary key returned per page.

Due to these limitations with Microsoft Dynamics GP, it is not possible to design a 100% accurate paging mechanism. To give you more flexibility over this problem, we have exposed the "PagingColumn" pseudo-column. It can be used to specify an alternative column to use for paging on like so:

```
SELECT * FROM PurchaseInvoice WHERE PagingColumn='Date'
```

Note that only the columns Microsoft Dynamics GP can handle as a criteria are available to be selected as a paging column.

Read-only Support

Adding support for reading metadata automatically and rewriting a most of the provider took a lot of development time. Due to an increasing number of users expecting our products to work in analytics tools, we prioritized making selection of data work seemlessly as much as possible. Due to time constraints, were unable to add back support for insert/update/delete operations for the release of the 2017 version. If you require insert/update/delete support, please contact us at support@cdata.com. If we see that more users need these features, we will prioritize adding them back to the Microsoft Dynamics GP Adapter.

Advanced Settings

The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

HTTP Proxies

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The Microsoft Dynamics GP Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Microsoft Dynamics GP API.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Microsoft Dynamics GP adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
}

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | IN | NOT
IN | AND } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Customer
```

Rename a column:
```
SELECT "Name" AS MY_Name FROM Customer
```

Search data:
```
SELECT * FROM Customer WHERE Name = 'Jon Doe';
```

The Microsoft Dynamics GP APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, IN, NOT IN, AND.
```
SELECT * FROM Customer WHERE Name = 'Jon Doe';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):
```
SELECT Id, Name INTO "csv://Customer.txt" FROM "Customer" WHERE
Name = 'Jon Doe'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

SELECT Id, Name INTO "csv://Customer.txt;delimiter=tab" FROM "Customer" WHERE Name = 'Jon Doe'

Data Model

The Microsoft Dynamics GP Adapter models Microsoft Dynamics GP entities in relational views, or read-only tables. The tables are determined automatically based on the metadata the adapter retrieves when you connect. Any changes that you make to your Microsoft Dynamics GP account, such as creating a custom field or changing its data type, are reflected on reconnection.

Views shows some sample view definitions included in the Microsoft Dynamics GP development environment. The actual views available will depend on your account.

A full list is available upon customer request.

TDV Microsoft Dynamics NAV Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to Microsoft Dynamics NAV

Before the adapter can connect with Microsoft Dynamics NAV, OData services need to be enabled on the server. Once OData Services are enabled, the adapter will be able to query any services that are published on the server.

In addition, specify a Url to a valid Microsoft Dynamics NAV server organization root (e.g. http://MyServer:7048) and a ServerInstance (e.g. DynamicsNAV71). If there is not a Service Default Company for the server, set the Company (e.g. 'CRONUS Canada, Inc.') as well.

In a multitenant installation, specify the tenant Id in Tenant (e.g. 'Cronus1').

Authenticating to Microsoft Dynamics NAV

To authenticate, set the User and Password properties to valid Microsoft Dynamics NAV logon credentials or Windows user credentials. Select the authentication method in AuthScheme.

Set AuthScheme to NEGOTIATE to authenticate using your Microsoft Dynamics NAV credentials. This is the default option.

Set AuthScheme to NTLM to perform authentication through Windows.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Scheme	The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, or NEGOTIATE. NEGOTIATE is the default option.
Company	The company to submit queries against. For example, 'CRONUS Canada, Inc.'.
Continue On Error	Whether or not to continue after encountering an error on a batch request.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password used to authenticate to the Dynamics NAV server.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.

Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to DynamicsNAV from the provider.
Server Instance	The instance of the Dynamics NAV server. For example, DynamicsNAV71.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Tenant	Use this value to connect to a specific Tenant in a multitenant installation of DynamicsNAV.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Url	URL to the DynamicsNAV server organization root. For example, http://MyServer:7048.
User	The user who is authenticating to the Dynamics NAV server.

Auth Scheme

The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, or NEGOTIATE. NEGOTIATE is the default option.

Data Type

string

Default Value

"NEGOTIATE"

Remarks

Together with Password and User, this field is used to authenticate against the server. NEGOTIATE is the default option. Use the following options to select your authentication scheme:

NTLM: Set this to use your Windows credentials to authenticate.

BASIC: Set this to use HTTP Basic authentication.

NEGOTIATE: If

AuthScheme is set to NEGOTIATE, the adapter will negotiate an authentication mechanism with the server. Set AuthScheme to NEGOTIATE to use Kerberos authentication.

DIGEST: Set this to use HTTP Digest authentication.

Company

The company to submit queries against. For example, 'CRONUS Canada, Inc.'.

Data Type

string

Default Value

""

Remarks

The company to submit queries against. For example, 'CRONUS Canada, Inc.'. This property is required if the Service Default Company has not been specified for the ServerInstance in question. If blank, the adapter will submit queries to the server and attempt to use the Service Default Company.

Continue On Error

Whether or not to continue after encountering an error on a batch request.

Data Type

bool

Default Value

false

Remarks

This connection property is only supported on servers with OData version 4.0 and higher. However, individual servers may choose to ignore this setting. Setting ContinueOnError to true will cause exceptions to be returned in the temporary table instead of being thrown when a batch request is attempted.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Microsoft Dynamics NAV and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password used to authenticate to the Dynamics NAV server.

Data Type

string

Default Value

""

Remarks

The password used to authenticate to the Dynamics NAV server.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to DynamicsNAV from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Server Instance

The instance of the Dynamics NAV server. For example, DynamicsNAV71.

Data Type

string

Default Value

""

Remarks

The instance of the Dynamics NAV server. For example, DynamicsNAV71.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer

The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Tenant

Use this value to connect to a specific Tenant in a multitenant installation of DynamicsNAV.

Data Type

string

Default Value

""

Remarks

Specify the tenant Id to connect to a specific tenant in a multitenant installation of DynamicsNAV. For example, "Cronus1".

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Url

URL to the DynamicsNAV server organization root. For example, http://MyServer:7048.

Data Type

string

Default Value

""

Remarks

URL to the Microsoft Dynamics NAV server organization root. For example, http://MyServer:7048.

User

The user who is authenticating to the Dynamics NAV server.

Data Type

string

Default Value

""

Remarks

The user who is authenticating to the Dynamics NAV server.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Microsoft Dynamics NAV Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

HTTP Proxies

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The Microsoft Dynamics NAV Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Microsoft Dynamics NAV API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Microsoft Dynamics NAV adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
```

```
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | <> | > | < | >= | <= | LIKE | IS NOT
NULL | IS NULL | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Customer
```

Rename a column:

```
SELECT "Contact" AS MY_Contact FROM Customer
```

Search data:

```
SELECT * FROM Customer WHERE Name <> 'CData, Inc.';
```

The Microsoft Dynamics NAV APIs support the following operators in the WHERE clause: =, !=, <>, >, <, >=, <=, LIKE, IS NOT NULL, IS NULL, AND, OR.

```
SELECT * FROM Customer WHERE Name <> 'CData, Inc.';
```

Sort a result set in ascending order:

```
SELECT No, Contact FROM Customer  ORDER BY Contact ASC
```

Predicate Functions

CEILING(double_expression)

Returns the double_expression rounded up to the nearest whole number (no decimal component).

double_expression: The double expression to round.

CONCAT(string_expr1, string_expr2)

Returns the string that is the concatenation of string_expr1 and string_expr2.

string_expr1: The first string to be concatenated.

string_expr2: The second string to be concatenated.

DAY(datetime_date)

Returns the integer that specifies the day component of the specified date.

datetime_date: The datetime string that specifies the date.

ENDSWITH(string_expression, string_suffix)

Returns true if string_expression ends with string_suffix, otherwise returns false.

string_expression: The string expression to search within.

string_suffix: The string suffix to search for.

FLOOR(double_expression)

Returns the double_expression rounded down to the nearest whole number (no decimal component).

double_expression: The double expression to round.

HOUR(datetime_time)

Returns the integer that specifies the hour component of the specified time.

datetime_time: The datetime string that specifies the time.

INDEXOF(string_expression, string_search)

Returns the index location where string_search is contained within string_expression.

string_expression: The string expression to search within.

string_search: The search value to locate within string_expression.

LENGTH(string_expression)

Returns the number of characters of the specified string expression.

string_expression: The string expression.

MINUTE(datetime_time)

Returns the integer that specifies the minute component of the specified time.

datetime_time: The datetime string that specifies the time.

MONTH(datetime_date)

Returns the integer that specifies the month component of the specified date.

datetime_date: The datetime string that specifies the date.

ROUND(double_expression)

Returns the double_expression to the nearest whole number (no decimal component).

double_expression: The double expression to round.

SECOND(datetime_time)

Returns the integer that specifies the second component of the specified time.

datetime_time: The datetime string that specifies the time.

STARTSWITH(string_expression, string_prefix)

Returns true if string_expression starts with string_prefix, otherwise returns false.

string_expression: The string expression to search within.

string_prefix: The string prefix to search for.

TOLOWER(string_expression)

Returns the string_expression with the uppercase character data converted to lowercase.

string_expression: The string expression to lowercase.

TOUPPER(string_expression)

Returns the string_expression with the lowercase character data converted to uppercase.

string_expression: The string expression to uppercase.

TRIM(string_expression)

Returns the string_expression with the leading and trailing whitespace removed.

string_expression: The string expression to trim.

YEAR(datetime_date)

Returns the integer that specifies the year component of the specified date.

datetime_date: The datetime string that specifies the date.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT No, Contact INTO "csv://Customer.txt" FROM "Customer" WHERE
Name = 'CData, Inc.'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT No, Contact INTO "csv://Customer.txt;delimiter=tab" FROM
"Customer" WHERE Name = 'CData, Inc.'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Customer (Contact) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Customer SET Contact='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1,
"http://myserver:7048/DynamicsNAV71/OData/Customer('01234567')");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:dynamicsnav:URL=http://myserver:
7048;User=myserver\Administrator;Password=admin;ServerInstance=DYN
AMICSNAV71;",);
String cmd = "DELETE FROM Customer WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1,
"http://myserver:7048/DynamicsNAV71/OData/Customer('01234567')");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The Microsoft Dynamics NAV Adapter models Microsoft Dynamics NAV entities in relational tables and stored procedures. The table definitions are dynamically obtained from the OData service you connect to.

A full list is available upon customer request.

TDV Amazon DynamoDB Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to DynamoDB

To authorize Amazon DynamoDB requests, provide the credentials for an administrator account or for an IAM user with custom permissions: Set AccessKey to the access key Id. Set SecretKey to the secret access key.

Note: Though you can connect as the AWS account administrator, it is recommended to use IAM user credentials to access AWS services.

Obtaining the Access Key

To obtain the credentials for an IAM user, follow the steps below:

Sign into the IAM console.

In the navigation pane, select Users.

To create or manage the access keys for a user, select the user and then select the Security Credentials tab.

To obtain the credentials for your AWS root account, follow the steps below:

Sign into the AWS Management console with the credentials for your root account.

Select your account name or number and select My Security Credentials in the menu that is displayed.

Click Continue to Security Credentials and expand the Access Keys section to manage or create root account access keys.

Authenticating from an EC2 Instance

If you are using the Amazon DynamoDB Adapter from an EC2 Instance and have an IAM Role assigned to the instance, you can use the IAM Role to authenticate. To do so, set UseEC2Roles to true and leave AccessKey and SecretKey empty. The Amazon DynamoDB Adapter will automatically obtain your IAM Role credentials and authenticate with them.

Authenticating as an AWS Role

In many situations it may be preferable to use an IAM role for authentication instead of the direct security credentials of an AWS root user. An AWS role may be used instead by specifying the RoleARN. This will cause the Amazon DynamoDB Adapter to attempt to retrieve credentials for the specified role. If you are connecting to AWS (instead of already being connected such as on an EC2 instance), you must additionally specify the AccessKey and SecretKey of an IAM user to asume the role for. Roles may not be used when specifying the AccessKey and SecretKey of an AWS root user.

Authenticating with MFA

For users and roles that require Multi-factor Authentication, specify the MFASerialNumber and MFAToken connection properties. This will cause the Amazon DynamoDB Adapter to submit the MFA credentials in a request to retrieve temporary authentication credentials. Note that the duration of the temporary credentials may be controlled via the TemporaryTokenDuration (default 3600 seconds).

Connecting to DynamoDB

In addition to the AccessKey and SecretKey properties, you can optionally set Domain and Region. Set Region to the region where your Amazon DynamoDB data is hosted. Set Domain if you want to use a domain name you have associated with AWS.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Access Key	Your AWS account access key. This value is accessible from your AWS security credentials page.
Auto Detect Index	A boolean indicating if secondary indexes should be automatically detected based on the query used.
Domain	Your AWS domain name. You can optionally choose to associate your domain name with AWS.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Flatten Arrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
Flatten Objects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
Generate Schema Files	Indicates the user preference as to when schemas should be generated and saved.
Insert Mode	How to handle values when inserting if the same primary key combination already exists in DynamoDB.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Maximum Request Retries	The maximum number of times to retry a request.

MFA Serial Number	The serial number of the MFA device if one is being used.
MFA Token	The temporary token available from your MFA device.
Number Column Mode	Specifies how to handle detected number columns. DynamoDB returns number values with a total precision of 38.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The maximum number of results to return per page from DynamoDB per request.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to DynamoDB from the provider.
Region	The hosting region for your Amazon Web Services.
Retry Wait Time	The minimum number of seconds the provider will wait to retry a request.
Role ARN	The optional Amazon Resource Name of the role to use when authenticating.
Row Scan Depth	The maximum number of rows to scan to look for the columns available in a table.

Secret Key	Your AWS account secret key. This value is accessible from your AWS security credentials page.
Separator Character	The character or characters used to denote hierarchy.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Temporary Token Duration	The amount of time (in seconds) a temporary token will last.
Thread Count	The number of threads to use when selecting data via a parallel scan. Setting ThreadCount to 1 will disable parallel scans.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Type Detection Scheme	Determines how to determine the data type of columns.
Url	The URL to use when submitting requests. This should not normally need to be set.
Use EC2 Roles	A boolean indicating if you would like to use an EC2 credentials.
Use Simple Names	Boolean determining if simple names should be used for tables and columns.

Access Key

Your AWS account access key. This value is accessible from your AWS security credentials page.

Data Type

string

Default Value

""

Remarks

Your AWS account access key. This value is accessible from your AWS security credentials page:

Sign into the AWS Management console with the credentials for your root account.

Select your account name or number and select My Security Credentials in the menu that is displayed.

Click Continue to Security Credentials and expand the Access Keys section to manage or create root account access keys.

Auto Detect Index

A boolean indicating if secondary indexes should be automatically detected based on the query used.

Data Type

bool

Default Value

true

Remarks

In DynamoDB, secondary indexes can be used to more quickly select data from a given table. By default, we attempt to automatically detect an index to use based on the query. However, this may not always be desirable. If you have control over the query and would prefer to specify the index yourself, set AutoDetectIndex to false and simply use the SecondaryIndex pseudo column to specify which index to use (if any).

Domain

Your AWS domain name. You can optionally choose to associate your domain name with AWS.

Data Type

string

Default Value

"amazonaws.com"

Remarks

If you do not have a unique AWS domain name, leave this value specified as amazonaws.com.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description

TUNNEL	80	When this is set, the adapter opens a connection to Amazon DynamoDB and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Flatten Arrays

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.

Data Type

string

Default Value

""

Remarks

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

Set FlattenArrays to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

For example, you can return an arbitrary number of elements from an array of strings:

```
["FLOW-MATIC","LISP","COBOL"]
```

When FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
languages_0	FLOW-MATIC

Flatten Objects

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Data Type

bool

Default Value

true

Remarks

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON. The property name is concatenated onto the object name with an underscore to generate the column name.

For example, you can flatten the nested objects below at connection time:

```
[
     { "grade": "A", "score": 2 },
     { "grade": "A", "score": 6 },
     { "grade": "A", "score": 10 },
     { "grade": "A", "score": 9 },
     { "grade": "B", "score": 14 }
]
```

When FlattenObjects is set to true and FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
grades_0_grade	A
grades_0_score	2

Generate Schema Files

Indicates the user preference as to when schemas should be generated and saved.

Data Type

string

Default Value

"Never"

Remarks

GenerateSchemaFiles enables you to save the table definitions identified by Automatic Schema Discovery. This property outputs schemas to .rsd files in the path specified by Location.

Available settings are the following:

Never: A schema file will never be generated.

OnUse: A schema file will be generated the first time a table is referenced, provided the schema file for the table does not already exist.

OnStart: A schema file will be generated at connection time for any tables that do not currently have a schema file.

Note that if you want to regenerate a file, you will first need to delete it.

Generate Schemas with SQL

When you set GenerateSchemaFiles to **OnUse**, the adapter generates schemas as you execute SELECT queries. Schemas are generated for each table referenced in the query.

Generate Schemas on Connection

Another way to use this property is to obtain schemas for every table in your database when you connect. To do so, set GenerateSchemaFiles to **OnStart** and connect.

Editing Schemas

Schema files have a simple format that makes them easy to modify. See Custom Schema Definitions for more information.

Insert Mode

How to handle values when inserting if the same primary key combination already exists in DynamoDB.

Data Type

string

Default Value

"REPLACE"

Remarks

If InsertMode is set to DONTREPLACE, an exception will be thrown from DynamoDB if you attempt to insert a value where the primary key combination already exists in DynamoDB. By default there will be no exception and the item in DynamoDB will be replaced by the newly inserted value.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Maximum Request Retries

The maximum number of times to retry a request.

Data Type

string

Default Value

""

Remarks

MaximumRequestRetries is the maximum number of times the adapter will retry a request when the problem has been detected as temporary (errors like "unknown error", network issues, and exceeding the maximum threshold per table). In this case on the first retry the adapter will back off and wait for the amount of time designated by RetryWaitTime. If that request fails, the adapter will double the time and then double again until the adapter has exhausted the available retries.

For example, if RetryWaitTime is set to 2 seconds and MaximumRequestRetries is set to 5, the wait times will be as follows: 0 -> 2 -> 4 -> 8 -> 16 -> 32.

MFA Serial Number

The serial number of the MFA device if one is being used.

Data Type

string

Default Value

""

Remarks

You can find the device for an IAM user by going to the AWS Management Console and viewing the user's security credentials. For virtual devices, this is actually an Amazon Resource Name (such as arn:aws:iam::123456789012:mfa/user).

MFA Token

The temporary token available from your MFA device.

Data Type

string

Default Value

""

Remarks

If MFA is required, this value will be used along with the MFASerialNumber to retrieve temporary credentials to login. The temporary credentials available from AWS will only last up to 1 hour. Once the time is up, the connection must be updated to specify a new MFA token so that new credentials may be obtained.

Number Column Mode

Specifies how to handle detected number columns. DynamoDB returns number values with a total precision of 38.

Data Type

string

Default Value

"DOUBLE"

Remarks

By default, all numbers will be specified as double. However, depending on your needs, you may wish to handle number columns as either double or decimal. In addition, you can also set the value to string to report all number columns as strings if you are using the Amazon DynamoDB Adapter in an environment that cannot handle the maximum precision size of DynamoDB.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The maximum number of results to return per page from DynamoDB per request.

Data Type

string

Default Value

""

Remarks

The Pagesize property affects the maximum number of results to return per page from DynamoDB. By default, DynamoDB will return up to 1MB of data per page. The Pagesize indicates the maximum number of items to return per page. DynamoDB may return fewer than this number if the total size of the items exceeds 1MB before hitting it.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to DynamoDB from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Region

The hosting region for your Amazon Web Services.

Data Type

string

Default Value

"NORTHERNVIRGINIA"

Remarks

The hosting region for your Amazon Web Services. Available values are NORTHERNVIRGINIA, OHIO, NORTHERNCALIFORNIA, OREGON, CENTRAL, IRELAND, FRANKFURT, LONDON, SINGAPORE, SYDNEY, SEOUL, TOKYO, MUMBAI, SAOPAULO, and BEIJING.

Retry Wait Time

The minimum number of seconds the provider will wait to retry a request.

Data Type

string

Default Value

""

Remarks

The value of this property is doubled on every retry to determine how long to wait until the next retry. Specify the maximum number of retries with MaximumRequestRetries.

Role ARN

The optional Amazon Resource Name of the role to use when authenticating.

Data Type

string

Default Value

""

Remarks

When authenticating outside of AWS, it is common to use a Role for authentication instead of your direct AWS account credentials. Entering the RoleARN will cause the Amazon DynamoDB Adapter to perform a role based authentication instead of using the AccessKey and SecretKey directly. The AccessKey and SecretKey must still be specified to perform this authentication. You cannot use the credentials of an AWS root user when setting RoleARN. The AccessKey and SecretKey must be those of an IAM user.

Row Scan Depth

The maximum number of rows to scan to look for the columns available in a table.

Data Type

string

Default Value

"50"

Remarks

Since DynamoDB is schemaless, the columns in a table must be determined by scanning table rows. This value determines the maximum number of rows that will be scanned.

See Table Columns for more details on the types discovered and how to define a static schema instead, an alternative that gives you more control over the projected columns.

Secret Key

Your AWS account secret key. This value is accessible from your AWS security credentials page.

Data Type

string

Default Value

""

Remarks

Your AWS account secret key. This value is accessible from your AWS security credentials page:

Sign into the AWS Management console with the credentials for your root account.

Select your account name or number and select My Security Credentials in the menu that is displayed.

Click Continue to Security Credentials and expand the Access Keys section to manage or create root account access keys.

Separator Character

The character or characters used to denote hierarchy.

Data Type

string

Default Value

"."

Remarks

In order to flatten out structures such as Maps and List attributes in DynamoDB, we need some specifier that states what the separation is between those columns and other columns. If this value is "." and a column comes back with the name address.city, this indicates that there is a mapped attribute with a child called city. If your data has columns that already use a single period within the attribute name, set the SeparatorCharacter to a different character or characters.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Temporary Token Duration

The amount of time (in seconds) a temporary token will last.

Data Type

string

Default Value

"3600"

Remarks

Temporary tokens are used with both MFA and Role based authentication. Temporary tokens will eventually time out, at which time a new temporary token must be obtained. For situations where MFA is not used, this is not a big deal. The Amazon DynamoDB Adapter will internally request a new temporary token once the temporary token has expired.

However, for MFA required connection, a new MFAToken must be specified in the connection to retrieve a new temporary token. This is a more intrusive issue since it requires an update to the connection by the user. The maximum and minimum that can be specified will depend largely on the connection being used.

For Role based authentication, the minimum duration is 900 seconds (15 minutes) while the maximum if 3600 (1 hour). Even if MFA is used with role based authentication, 3600 is still the maximum.

For MFA authentication by itself (using an IAM User or root user), the minimum is 900 seconds (15 minutes), the maximum is 129600 (36 hours).

Thread Count

The number of threads to use when selecting data via a parallel scan. Setting ThreadCount to 1 will disable parallel scans.

Data Type

string

Default Value

"4"

Remarks

Parallel scans allow data to be retrieved faster by splitting up the retrieval process across multiple threads. This can greatly improve performance when scanning data in Amazon DynamoDB. However, this will also consume your read units for a table much faster than a single thread. Consider your available cores, bandwidth, and read units for your tables before increasing the ThreadCount.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Type Detection Scheme

Determines how to determine the data type of columns.

Data Type

string

Default Value

""

Remarks

None	Setting TypeDetectionScheme to None will return all columns as string type. Note: Even when set to None, the column names will still be scanned when Header=True.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned. If no value is specified, RowScan will be used by default.

Url

The URL to use when submitting requests. This should not normally need to be set.

Data Type

string

Default Value

""

Remarks

The URL will normally be determined automatically based on your Region and Domain. However, you can optionally specify the URL as an override. For instance, this can work for connecting to Local DynamoDB.

Use EC2 Roles

A boolean indicating if you would like to use an EC2 credentials.

Data Type

bool

Default Value

false

Remarks

This property takes advantage of using EC2 roles. If the Amazon DynamoDB Adapter is running on an EC2 instance and the instance has an IAM Role assigned to it, the role can be used without any other credentials specified. When UseEC2Roles is set to true, other authentication credentials will be ignored. This connection property should not be used when connecting from outside of an EC2 instance.

Use Simple Names

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

Boolean determining if simple names should be used for tables and columns. DynamoDB tables and columns can use special characters in names that are normally not allowed in standard databases. UseSimpleNames makes the adapter easier to use with traditional database tools.

Setting UseSimpleNames to true will simplify the names of tables and columns returned. It will enforce a naming scheme such that only alphanumeric characters and the underscore are valid for displayed table and column names. Any non-alphanumeric characters will be converted to an underscore.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Amazon DynamoDB Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Querying The Data

DynamoDB has two distinct operations to read data from tables: query and scan. The Amazon DynamoDB Adapter will automatically attempt to figure out which type to use based on the WHERE clause of the SQL statement.

Query vs. Scan

In DynamoDB, a query is used when some of the data can be filtered before results are returned. This is done by the use of partition keys and sort keys that are defined on the table to perform the filter. It is typically much faster than a scan.

A scan is performed when anything other than a partition key or a sort key is used to filter the data. In the case of the scan, data is only filtered after it is returned. This is much slower since it will cause DynamoDB to return everything and simply exclude nonmatching results from the response. In the worst cases it will return full pages of data that are simply empty due to no matching results.

Secondary Indexes

Secondary indexes in DynamoDB behave like tables in themselves that are only accessible off of a given table. A secondary index behaves like a view either with predefined columns or simply with all columns being accessible. The main difference is that different partition and sort keys may be selected for them. There is no restriction on which keys to use - they may be the same as the partition and sort keys for the table itself, the same as ones from a different secondary index, or completely unique to that index.

The adapter by default will attempt to determine if a secondary index should be used to perform a query operation. This is done by analyzing the WHERE clause of the SQL statement. If it is determined that a query should be used and that a specific index should be used for that query, then the secondary index will automatically be selected. This behavior can be disabled by setting AutoDetectIndex to false.

A secondary index can be explicitly specified by using the SecondaryIndex pseudo column. This will override AutoDetectIndex and will be used for both query and scan operations.

Query Examples

A query operation will be submitted to DynamoDB if the WHERE clause includes a single PartitionKey with an '=' operator and optionally up to one SortKey using any DynamoDB supported operator. For instance:

```
SELECT * FROM Table WHERE PartitionKey = 'x'
```

```
SELECT * FROM Table WHERE PartitionKey = 'x' AND SortKey > 'y' AND
SortKey < 'z'
```

Additionally, if AutoDetectIndex is set to true, the adapter will attempt to automatically determine an index based on the specific keys used. For instance:

```
SELECT * FROM Table WHERE SecondaryIndexPartitionKey = 'x'
```

The preceding query would result in a query against the SecondaryIndex, provided that it was the only one that used the specified PartitionKey.

A secondary index may be explicitly specified via the SecondaryIndex pseudo column. This will cause the specified index to be used regardless of if a Query or Scan is submitted:

```
SELECT * FROM Table WHERE SecondaryIndex = 'IndexName'
```

Parallel Scans

For better performance when scanning results, parallel scans may be used. Parallel scans are a way to use multiple threads to execute the same SELECT statement. Each thread is responsible for retrieving an equal share of the data. Because threads retrieve data independently, it can exponentially improve performance. However, performance gains using this method will be limited to CPU performance, number of cores, bandwidth constraints, and read units for the table. A machine with more cores and more bandwidth may benefit from using additional threads whereas a machine with fewer cores and less bandwith available will see no benefit.

Amazon DynamoDB offers no equivalent for parallel scans when querying results. Queries can only be executed in a single thread.

By default, parallel scans are enabled. They can be disabled or changed by updating the ThreadCount connection property.

Please be aware that using parallel scans will consume your allocated read units for a given table at a much faster rate than using a single thread. If you are getting throttle exceptions, it is recommended to decrease the ThreadCount or set it to 1.

Advanced Settings

Accessing NoSQL Tables

The adapter implements Automatic Schema Discovery that is highly configurable. The following sections outline the adapter's defaults and link to ways to further customize.

Flattening Nested JSON

By default, the adapter projects columns over the properties of objects. Arrays are returned as JSON strings, by default. You can use the following properties to access array elements, including objects nested in arrays.

FlattenArrays: Set this property to the number of array elements that you want to return as column values. You can also use this property with FlattenObjects to extract the properties of objects nested in arrays.

FlattenObjects: By default, this is true; that is, the properties of objects and nested objects are returned as columns. When you set FlattenArrays, objects nested in the specified array elements are also flattened and returned as columns.

Other mechanisms for accessing nested objects are detailed in NoSQL Database.

Inferring the Data Type

You can use the following properties to configure automatic data type detection, which is enabled by default.

TypeDetectionScheme: You can use this property to enable or disable automatic type detection based on the value specified in RowScanDepth.

RowScanDepth: This property determines the number of rows that will be scanned to determine column data types.

Fine Tuning Data Access

You can use the following properties to gain greater control over Amazon DynamoDB API features and the strategies the adapter uses to surface them:

GenerateSchemaFiles: This property enables you to persist table metadata in static schema files that are easy to customize, to change column data types, for example. You can set this property to "OnStart" to generate schema files for all tables in your database at connection. Or, you can generate schemas as you execute SELECT queries to tables. The resulting schemas are based on the connection properties you use to configure Automatic Schema Discovery

.

NumberColumnMode: This property enables you to specify an SQL data type for the Amazon DynamoDB "number" type, such as double, decimal, or string.

InsertMode: Inserts in Amazon DynamoDB have several key differences from SQL inserts. An insert in Amazon DynamoDB is an upsert; that is, a row that already exists is replaced by the incoming row. Also, the primary key must be specified.

By default, the adapter will perform an upsert based on the primary key. You can use this connection property to throw an error instead of overwriting existing records.

To use the resulting schema files, set the Location property to the folder containing the schemas.

UseSimpleNames: Amazon DynamoDB supports attribute names with special characters that many database-oriented tools do not support.

In addition, Amazon DynamoDB table names can include dots and dashes -- the adapter interprets dots within table names as hierarchy separators that enable you to drill down to nested fields, similar to XPath.

You can use this property to replace any nonalphanumeric character with an underscore.

SeparatorCharacter: You can use this property to more easily access nested fields when Querying Documents and Lists

; specify the hierarchy separator with this property. By default, this character is the '.' (dot) character.

Fine Tuning Performance

You can use the following connection properties to avoid "maximum throughput exceeded" errors.

Using Paging Effectively

You can use the Pagesize property to optimize use of your provisioned throughput, based on the size of your items and Amazon DynamoDB's 1MB page size. Set this property to the number of items to return.

Generally, a smaller page size reduces spikes in throughput that cause throttling. A smaller page size also inserts pauses between requests. This interval evens out the distribution of requests and allows more requests to be successful by avoiding throttling.

You can also take advantage of secondary indexes by specifying them in the WHERE clause.

You can specify partition keys and sort keys in the WHERE clause to improve the performance of query operations. Based on these inputs, the adapter automatically detects the secondary index for a query operation.

See Querying The Data for more information on scan operations vs. query operations.

Setting a Retry Interval

You can set the following properties to retry queries instead of throwing a temporary error such as "maximum throughput exceeded":

RetryWaitTime: The minimum number of seconds the adapter will wait to retry a request.

MaximumRequestRetries: The maximum number of times to retry a request.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the Amazon DynamoDB authentication properties and, if necessary, Region and Domain. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

NoSQL Database

Amazon DynamoDB is a schemaless database that provides high performance, availability, and scalability. These features are not necessarily incompatible with a standards-compliant query language like SQL-92. In this section we will show various schemes that the adapter offers to bridge the gap with relational SQL and a document database.

The adapter models the schemaless Amazon DynamoDB tables into relational tables and translates SQL queries into Amazon DynamoDB queries to get the requested data. The adapter offers two ways, Automatic Schema Discovery and Custom Schema Definitions, to model Amazon DynamoDB tables as relational tables.

The Automatic Schema Discovery scheme automatically finds the data types in a Amazon DynamoDB table by scanning a configured number of rows of the table. You can use RowScanDepth, FlattenArrays, and FlattenObjects to control the relational representation of the tables in Amazon DynamoDB.

Optionally, you can use Custom Schema Definitions to project your chosen relational structure on top of a Amazon DynamoDB table. This allows you to define your chosen column names, their data types, and the location of their values in the Amazon DynamoDB table.

Automatic Schema Discovery

The adapter automatically infers a relational schema by inspecting a series of Amazon DynamoDB items in a table. You can use the RowScanDepth property to define the number of items the adapter will scan to do so. The columns identified during the discovery process depend on the FlattenArrays and FlattenObjects properties.

If FlattenObjects is set, all nested objects will be flattened into a series of columns. For example, consider the following item:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
```

```
  address: {street: "Main Street", city: "Chapel Hill", state:
"NC"},
  offices: ["Chapel Hill", "London", "New York"],
  annual_revenue: 35600000
}
```

This document will be represented by the following columns:

Column Name	Data Type	Example Value
id	Integer	12
name	String	Lohia Manufacturers Inc.
address_street	String	Main Street
address_city	String	Chapel Hill
address_state	String	NC
offices	String	["Chapel Hill", "London", "New York"]
annual_revenue	Double	35,600,000

If FlattenObjects is not set, then the address_string, address_city, and address_state columns will not be broken apart. The address column of type string will instead represent the entire object. Its value would be {street: "Main Street", city: "Chapel Hill", state: "NC"}. See JSON Functions for more details on working with JSON aggregates.

The FlattenArrays property can be used to flatten array values into columns of their own. This is only recommended for arrays that are expected to be short, for example the coordinates below:

```
"coord": [ -73.856077, 40.848447 ]
```

The FlattenArrays property can be set to 2 to represent the array above as follows:

Column Name	Data Type	Example Value
coord_0	Float	-73.856077

coord_1	Float	40.848447

It is best to leave other unbounded arrays as they are and piece out the data for them as needed using JSON Functions.

Vertical Flattening

It is possible to retrieve an array of objects as if it were a separate table. Take the following JSON structure from the restaurants collection for example:

```
{
  "restaurantid" : "30075445",
  "address" : {
    "building" : "1007",
    "coord" : "-73.856077, 40.848447",
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : "{
      "date" : 1393804800000,
      "grade" : "A",
      "score" : 2
    }, {
      "date" : 1378857600000,
      "grade" : "A",
      "score" : 6
    }, {
      "date" : 1358985600000,
      "grade" : "A",
      "score" : 10
    }, {
      "date" : 1322006400000,
      "grade" : "A",
      "score" : 9
    }, {
      "date" : 1299715200000,
      "grade" : "B",
      "score" : 14
    }",
  "name" : "Morris Park Bake Shop",
}
```

Vertical flattening will allow you to retrieve the grades array as a separate table:

```
SELECT * FROM "restaurants.grades"
```

This query returns the following data set:

date	grade	score	restaurantid	_index
1393804800000	A	2	30075445	1
1378857600000	A	6	30075445	2
1358985600000	A	10	30075445	3

You may also want to include information from the base restaurants table. You can do this with a join.

```
SELECT restaurants.name, "restaurants.grades".* FROM "restaurants"
JOIN "restaurants.grades" ON restaurants.restaurantid =
"restaurants.grades".restaurantid WHERE restaurants.restaurantid =
30075445 AND "restaurants.grades".restaurantid = 30075445
```

This query returns the following data set:

name	restaurantid	date	grade	score	_index
Morris Park Bake Shop	30075445	1393804800000	A	2	1
Morris Park Bake Shop	30075445	1378857600000	A	6	2
Morris Park Bake Shop	30075445	1358985600000	A	10	3
Morris Park Bake Shop	30075445	1322006400000	A	9	4
Morris Park Bake Shop	30075445	1299715200000	B	14	5

JSON Functions

The adapter can return JSON structures as column values. The adapter enables you to use standard SQL functions to work with these JSON structures. The examples in this section use the following array:

```
[
```

```
    { "grade": "A", "score": 2 },
    { "grade": "A", "score": 6 },
    { "grade": "A", "score": 10 },
    { "grade": "A", "score": 9 },
    { "grade": "B", "score": 14 }
]
```

JSON_EXTRACT

The JSON_EXTRACT function can extract individual values from a JSON object. The following query returns the values shown below based on the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_EXTRACT(grades,'[0].grade') AS Grade,
JSON_EXTRACT(grades,'[0].score') AS Score FROM Students;
```

Column Name	Example Value
Grade	A
Score	2

JSON_COUNT

The JSON_COUNT function returns the number of elements in a JSON array within a JSON object. The following query returns the number of elements specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_COUNT(grades,'[x]') AS NumberOfGrades FROM
Students;
```

Column Name	Example Value
NumberOfGrades	5

JSON_SUM

The JSON_SUM function returns the sum of the numeric values of a JSON array within a JSON object. The following query returns the total of the values specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_SUM(score,'[x].score') AS TotalScore FROM
Students;
```

Column Name	Example Value
TotalScore	41

JSON_MIN

The JSON_MIN function returns the lowest numeric value of a JSON array within a JSON object. The following query returns the minimum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MIN(score,'[x].score') AS LowestScore FROM
Students;
```

Column Name	Example Value
LowestScore	2

JSON_MAX

The JSON_MAX function returns the highest numeric value of a JSON array within a JSON object. The following query returns the maximum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MAX(score,'[x].score') AS HighestScore FROM
Students;
```

Column Name	Example Value
HighestScore	14

DynamoDB Queries

Because Amazon DynamoDB is a NoSQL data source, queries need to be handled a bit differently than standard relational databases.

Value-Sensitive Queries

The lack of a required data type for a given column means that you could store different types of data in a single column. For instance, one row could have a String called EmailAddresses and another could have a StringSet also called EmailAddresses. For these and other kinds of cases, the adapter largely determines what data type to use based on the values in the query.

For instance, say you have an Items table where the PartNumber could store either a String or a Number. To get back a part with the PartNumber of the number value 12345, you would issue the following query:

```
SELECT Name, Location, Quantity, PartNumber FROM Items WHERE
PartNumber = 12345
```

Alternatively, the PartNumber could have been stored as the string "12345". To get back a part with the PartNumber of the literal string 12345, issue the following query:

```
SELECT Name, Location, Quantity, PartNumber FROM Items WHERE
PartNumber = '12345'
```

If the data type of the specified value is not ambiguous, it is always used before the autodetected data type. In both of these cases if a parameter was used instead of of a hardcoded value, then the data type of the parameter would be used to determine what type to submit to Amazon DynamoDB.

Detected Column Data Type

If a value is not obvious based purely on the detected data type, the adapter compares it to the autodetected column. For instance, if you want to insert a column called Coordinates into the Location table, your insert would look like:

```
INSERT INTO Locations (Address, Coordinates) VALUES ('123 Fake
Street', '[40.7127, 74.0059]')
```

Based on the input value alone, the detected data type is a string. However, because a Coordinates column was previously autodetected, the adapter inserts a NumberSet and not a simple String.

If a Coordinates column was not autodetected when scanning the Locations table, the data type of the inserted value is used.

In this case, we could still resolve that the insert is a NumberSet, but it will cost a bit more overhead to do this.

Querying Documents and Lists

Amazon DynamoDB documents and lists are supported with the Amazon DynamoDB Adapter. You can access documents and lists directly at the root level or use the '.' character as a hierarchy divider to drill down to documents and lists.

Reporting Values in Documents and Lists

When data types are autodetected, they are reported down to the lowest level that can be reliably detected. For instance, a document called Customer with a child called Address and a child on Address called Street would be represented by the column Customer.Address.Street.

However, this process does not apply to Lists since a list could have any number of entries. Once a List or a Set is detected, additional values are not reported as being available in the table schema.

Getting Back Unreported Values

If there are attributes that frequently do not have a value and thus are not autodetected, these can still be retrieved by specifying the correct path to them. For instance, to get the Special attribute from the Customer document:

```
SELECT [Customer.Address.Street], [Customer.Special] FROM MyTable
```

Once a List has been detected, additional values are not reported. But individual values on the list can be referenced by specifying '.' and a number. For instance:

```
SELECT [MyList.0], [MyList.1.Email], [MyList.1.Age] FROM MyTable
```

This will retrieve the first value on the list and the second value's Email and Age attributes.

Inserting Documents and Lists

Inserts in Amazon DynamoDB require that the full object is specified during insert. Insert a document or list at the root. Pass in the full JSON aggregate. For instance:

```
INSERT INTO MyTable (PrimaryKey, EmailAddresses, Address, MyList)
VALUES ('uniquekey', '["user@email.com", "user2@email2.com"]',
'{"Street":"123 Fake Street", "City":"Chapel Hill",
"Zip":"27713"}', '[{"S":"somestr"},{"NS":[1,2]},{"N":4}]')
```

In this case, the EmailAddress is inserted as a StringSet, Address is inserted as a document, and MyList is inserted as a list.

Updating Documents and Lists

Updates are supported using the same syntax that is available during selects. Documents and Lists can be specified using the '.' character to specify hierarchy. For instance:

```
UPDATE MyTable SET [EmailAddress.0]='user@email.com',
[EmailAddress.1]='user2@email2.com', [Address.Street]='123 Fake
Street', [Address.City]='Chapel Hill', [Address.Zip]='27713',
[MyList.0]='somestr', [MyList.1]='[1,2]', [MyList.2]=4 WHERE
PrimaryKey='uniquekey'
```

Note that EmailAddress and MyList must be autodetected to resolve how to handle EmailAddress differently from MyList. If you are in doubt about whether or not something will be automatically detected, specifying the full JSON to update will always work.

Custom Schema Definitions

In addition to Automatic Schema Discovery the adapter also allows you to statically define the schema for your Amazon DynamoDB table. Let's consider a schema for the restaurants data set.

Below is an example item from the table:

```
{
   "address":{
      "building":"461",
      "coord":[
         -74.138492,
         40.631136
      ],
      "street":"Port Richmond Ave",
      "zipcode":"10302"
   },
   "borough":"Staten Island",
   "cuisine":"Other",
   "grades":[

   ],
   "name":"Indian Oven",
   "restaurant_id":"50018994"
}
```

Defining a Custom Schema

You can define a custom schema to extract out nested properties as their own columns. The following schema uses the other:bsonpath property to define where the data for a particular column should be retrieved from. Using this model you can flatten arbitrary levels of hierarchy.

The collection attribute specifies the collection to parse. The collection attribute gives you the flexibility to use multiple schemas for the same collection. If collection is not specified, the filename determines the collection that is parsed.

In Custom Schema Example, you will find the complete schema that contains the example above.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

  <rsb:info title="StaticRestaurants" description="Custom Schema
for the DynamoDB restaurants data set.">
    <!-- Column definitions -->
    <attr name="id"          xs:type="integer"
other:internaltype="N"    other:path="restaurant_id"
iskey="true"   other:keytype="HASH"  />
    <attr name="name"        xs:type="string"
other:internaltype="S"    other:path="name"                />
    <attr name="borough"     xs:type="string"
other:internaltype="S"    other:path="borough"             />
    <attr name="cuisine"     xs:type="string"
other:internaltype="S"    other:path="cuisine"            />
    <attr name="building"    xs:type="string"
other:internaltype="S"    other:path="address.building"    />
    <attr name="street"      xs:type="string"
other:internaltype="S"    other:path="address.street"      />
    <attr name="latitude"    xs:type="double"
other:internaltype="N"    other:path="address.coord.0"     />
    <attr name="longitude"   xs:type="double"
other:internaltype="N"    other:path="address.coord.1"     />
    <input name="rows@next" desc="Internal attribute used for
paging through data."/>
  </rsb:info>

  <rsb:set attr="collection" value="restaurants"/>

</rsb:script>
```

Custom Schema Example

In this section is a complete schema. The info section enables a relational view of a Amazon DynamoDB table. For more details, see Custom Schema Definitions. The table below allows the SELECT, INSERT, UPDATE, and DELETE commands as implemented in the GET, POST, MERGE, and DELETE sections of the schema below.

Use the tablename attribute to specify the name of the table you want to parse. You can use the tablename attribute to define multiple schemas for the same table.

If tablename is not specified, the filename determines the table that is parsed.

Use the pathseparator attribute to specify the character or characters to use to separate the fields in path to a given column. This will only be used when parsing the other:path input. If not specified, the default is the '.' character.

Note: Amazon DynamoDB is case sensitive. Your tablename and specified paths must match the case of how your fields appear in Amazon DynamoDB.

Copy the rows@next input as-is into your schema. The operations, such as dynamodbadoSelect, are internal implementations and can also be copied as is.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

  <rsb:info title="StaticRestaurants" description="Custom Schema
for the DynamoDB restaurants data set.">
    <!-- Column definitions -->
    <attr name="id"          xs:type="integer"
other:internaltype="N"    other:path="restaurant_id"
iskey="true"   other:keytype="HASH"  />
    <attr name="name"        xs:type="string"
other:internaltype="S"    other:path="name"                />
    <attr name="borough"     xs:type="string"
other:internaltype="S"    other:path="borough"             />
    <attr name="cuisine"     xs:type="string"
other:internaltype="S"    other:path="cuisine"             />
    <attr name="building"    xs:type="string"
other:internaltype="S"    other:path="address.building"    />
    <attr name="street"      xs:type="string"
other:internaltype="S"    other:path="address.street"      />
    <attr name="latitude"    xs:type="double"
other:internaltype="N"    other:path="address.coord.0"     />
    <attr name="longitude"   xs:type="double"
other:internaltype="N"    other:path="address.coord.1"     />
    <input name="rows@next" desc="Internal attribute used for
paging through data."/>
  </rsb:info>

  <rsb:set attr="tablename"       value="restaurants"/>
  <rsb:set attr="pathseparator"   value="."/>

  <rsb:script method="GET">
    <rsb:call op="dynamodbadoSelect">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

  <rsb:script method="POST">
    <rsb:call op="dynamodbadoAdd">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

  <rsb:script method="MERGE">
    <rsb:call op="dynamodbadoUpdate">
      <rsb:push/>
    </rsb:call>
  </rsb:script>
```

```
  <rsb:script method="DELETE">
    <rsb:call op="dynamodbadoDelete">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

</rsb:script>
```

System Tables

Query the following system tables to access schema information, information on data source functionality, and batch operation statistics.

Schema Tables

The following tables return database metadata for Amazon DynamoDB:

sys_catalogs

: Lists the available databases.

sys_schemas

: Lists the available schemas.

sys_tables

: Lists the available tables.

sys_tablecolumns

: Describes the columns of the available tables.

sys_views

: Lists the available views.

sys_viewcolumns

: Describes the columns of available views.

sys_procedures

: Describes the available stored procedures.

sys_procedureparameters

: Describes stored procedure parameters.

sys_keycolumns

: Describes the primary and foreign keys.

sys_indexes

: Describes the available indexes.

Data Source Tables

The following tables return information about how to connect to and query the data source:

sys_connection_props

: Returns information on the available connection properties.

sys_sqlinfo

: Describes the SELECT queries that the adapter can offload to the data source.

Query Information Tables

Query statistics are available for data modification queries, including batch operations.

sys_identity

: Returns information about batch operations or single updates.

Query the following system tables to access schema information, information on data source functionality, and batch operation statistics.

Schema Tables

The following tables return database metadata for Amazon DynamoDB:

sys_catalogs

: Lists the available databases.

sys_schemas

: Lists the available schemas.

sys_tables

: Lists the available tables.

sys_tablecolumns

: Describes the columns of the available tables.

sys_views

: Lists the available views.

sys_viewcolumns

: Describes the columns of available views.

sys_procedures

: Describes the available stored procedures.

sys_procedureparameters

: Describes stored procedure parameters.

sys_keycolumns

: Describes the primary and foreign keys.

sys_indexes

: Describes the available indexes.

Data Source Tables

The following tables return information about how to connect to and query the data source:

sys_connection_props

: Returns information on the available connection properties.

sys_sqlinfo

: Describes the SELECT queries that the adapter can offload to the data source.

Query Information Tables

Query statistics are available for data modification queries, including batch operations.

sys_identity

: Returns information about batch operations or single updates.

sys_catalogs

Lists the available databases.

The following query retrieves all databases determined by the connection string:

```
SELECT * FROM sys_catalogs
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.

Lists the available databases.

The following query retrieves all databases determined by the connection string:

```
SELECT * FROM sys_catalogs
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.

sys_schemas

Lists the available schemas.

```
SELECT * FROM sys_schemas
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.
SchemaName	*String*	The schema name.

Lists the available schemas.

```
SELECT * FROM sys_schemas
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.
SchemaName	*String*	The schema name.

sys_tables

Lists the available tables.

```
SELECT * FROM sys_tables
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table.
TableType	*String*	The table type.
Description	*String*	A description of the table.

Lists the available tables.

```
SELECT * FROM sys_tables
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table.
TableType	*String*	The table type.
Description	*String*	A description of the table.

sys_tablecolumns

Describes the columns of the available tables.

The following query returns the columns and data types for the Account table:

```
SELECT ColumnName, DataTypeName FROM sys_tablecolumns WHERE
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table containing the column.
ColumnName	*String*	The column name.
DataTypeName	*String*	The data type name.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	A brief description of the column.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

Describes the columns of the available tables.

The following query returns the columns and data types for the Account table:

```
SELECT ColumnName, DataTypeName FROM sys_tablecolumns WHERE
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table containing the column.
ColumnName	*String*	The column name.
DataTypeName	*String*	The data type name.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	A brief description of the column.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_views

Lists the available views.

```
SELECT TableName FROM sys_views
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
TableType	*String*	The type of the view.
Description	*String*	A description of the view.

Lists the available views.

```
SELECT TableName FROM sys_views
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
TableType	*String*	The type of the view.
Description	*String*	A description of the view.

sys_viewcolumns

Describes the columns of the available views.

The following query returns the columns and data types for a specified view:

```
SELECT ColumnName, DataTypeName FROM sys_viewolumns WHERE
TableName='MyView'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
ColumnName	*String*	The name of the column.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type of the column. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	The column description.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

Describes the columns of the available views.

The following query returns the columns and data types for a specified view:

```
SELECT ColumnName, DataTypeName FROM sys_viewolumns WHERE
TableName='MyView'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
ColumnName	*String*	The name of the column.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type of the column. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	The column description.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_procedures

Lists the available stored procedures.

```
SELECT * FROM sys_procedures
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the stored procedure.
SchemaName	*String*	The schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure.
Description	*String*	A description of the stored procedure.

Lists the available stored procedures.

```
SELECT * FROM sys_procedures
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the stored procedure.
SchemaName	*String*	The schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure.
Description	*String*	A description of the stored procedure.

sys_procedureparameters

Describes stored procedure parameters.

The following query returns information about all of the input parameters for the CreateTable stored procedure:

```
SELECT * FROM sys_procedureparameters WHERE
ProcedureName='CreateTable' AND Direction=1 OR Direction=2
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the stored procedure.
SchemaName	*String*	The name of the schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure containing the parameter.
ColumnName	*String*	The name of the stored procedure parameter.
Direction	*Int32*	An integer corresponding to the type of the parameter: input (1), input/output (2), or output(4). input/output type parameters can be both input and output parameters.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The number of characters allowed for character data. The number of digits allowed for numeric data.
NumericPrecision	*Int32*	The maximum precision for numeric data. The column length in characters for character and date-time data.
NumbericScale	*Int32*	The number of digits to the right of the decimal point in numeric data.
IsNullable	*Boolean*	Whether the parameter can contain null.
Description	*String*	The description of the parameter.
Ordinal	*Int32*	The index of the parameter.

Describes stored procedure parameters.

The following query returns information about all of the input parameters for the CreateTable stored procedure:

```
SELECT * FROM sys_procedureparameters WHERE
ProcedureName='CreateTable' AND Direction=1 OR Direction=2
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the stored procedure.
SchemaName	*String*	The name of the schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure containing the parameter.
ColumnName	*String*	The name of the stored procedure parameter.
Direction	*Int32*	An integer corresponding to the type of the parameter: input (1), input/output (2), or output(4). input/output type parameters can be both input and output parameters.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The number of characters allowed for character data. The number of digits allowed for numeric data.
NumericPrecision	*Int32*	The maximum precision for numeric data. The column length in characters for character and date-time data.
NumbericScale	*Int32*	The number of digits to the right of the decimal point in numeric data.
IsNullable	*Boolean*	Whether the parameter can contain null.
Description	*String*	The description of the parameter.
Ordinal	*Int32*	The index of the parameter.

sys_keycolumns

Describes the primary and foreign keys.

The following query retrieves the primary key for the Account table:

```
SELECT * FROM sys_keycolumns WHERE IsKey='True' AND
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the key.
SchemaName	*String*	The name of the schema containing the key.
TableName	*String*	The name of the table containing the key.
ColumnName	*String*	The name of the key column.
IsKey	*Boolean*	Whether the column is a primary key in the table referenced in the TableName table.
IsForeignKey	*Boolean*	Whether the column is a foreign key referenced in the TableName table.
ReferencedCatalogName	*String*	The database containing the primary key.
ReferencedSchemaName	*String*	The schema containing the primary key.
ReferencedTableName	*String*	The table containing the primary key.
ReferencedColumnName	*String*	The column name of the primary key.

Describes the primary and foreign keys.

The following query retrieves the primary key for the Account table:

```
SELECT * FROM sys_keycolumns WHERE IsKey='True' AND
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the key.
SchemaName	*String*	The name of the schema containing the key.
TableName	*String*	The name of the table containing the key.
ColumnName	*String*	The name of the key column.
IsKey	*Boolean*	Whether the column is a primary key in the table referenced in the TableName table.
IsForeignKey	*Boolean*	Whether the column is a foreign key referenced in the TableName table.
ReferencedCatalogName	*String*	The database containing the primary key.
ReferencedSchemaName	*String*	The schema containing the primary key.
ReferencedTableName	*String*	The table containing the primary key.
ReferencedColumnName	*String*	The column name of the primary key.

sys_indexes

Describes the available indexes. By filtering on indexes, you can write more selective queries with faster query response times.

The following query retrieves all indexes that are not primary keys:

```
SELECT * FROM sys_indexes WHERE IsPrimary='false'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the index.
SchemaName	*String*	The name of the schema containing the index.
TableName	*String*	The name of the table containing the index.
IndexName	*String*	The index name.
ColumnName	*String*	The name of the column associated with the index.
IsUnique	*Boolean*	True if the index is unique. False otherwise.
IsPrimary	*Boolean*	True if the index is a primary key. False otherwise.
Type	*Int16*	An integer value corresponding to the index type: statistic (0), clustered (1), hashed (2), or other (3).
SortOrder	*String*	The sort order: A for ascending or D for descending.
OrdinalPosition	*Int16*	The sequence number of the column in the index.

Describes the available indexes. By filtering on indexes, you can write more selective queries with faster query response times.

The following query retrieves all indexes that are not primary keys:

```
SELECT * FROM sys_indexes WHERE IsPrimary='false'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the index.
SchemaName	*String*	The name of the schema containing the index.
TableName	*String*	The name of the table containing the index.
IndexName	*String*	The index name.
ColumnName	*String*	The name of the column associated with the index.
IsUnique	*Boolean*	True if the index is unique. False otherwise.
IsPrimary	*Boolean*	True if the index is a primary key. False otherwise.
Type	*Int16*	An integer value corresponding to the index type: statistic (0), clustered (1), hashed (2), or other (3).
SortOrder	*String*	The sort order: A for ascending or D for descending.
OrdinalPosition	*Int16*	The sequence number of the column in the index.

sys_connection_props

Returns information on the available connection properties and those set in the connection string.

The following query retrieves all connection properties that have been set in the connection string or set through a default value:

```
SELECT * FROM sys_connection_props WHERE Value <> ''
```

Columns

Name	Type	Description
Name	*String*	The name of the connection property.
ShortDescription	*String*	A brief description.
Type	*String*	The data type of the connection property.
Default	*String*	The default value if one is not explicitly set.
Values	*String*	A comma-separated list of possible values. A validation error is thrown if another value is specified.
Value	*String*	The value you set or a preconfigured default.
Required	*Boolean*	Whether the property is required to connect.
Category	*String*	The category of the connection property.
IsSessionProperty	*String*	Whether the property is a session property, used to save information about the current connection.

Returns information on the available connection properties and those set in the connection string.

The following query retrieves all connection properties that have been set in the connection string or set through a default value:

```
SELECT * FROM sys_connection_props WHERE Value <> ''
```

Columns

Name	Type	Description
Name	*String*	The name of the connection property.
ShortDescription	*String*	A brief description.
Type	*String*	The data type of the connection property.
Default	*String*	The default value if one is not explicitly set.
Values	*String*	A comma-separated list of possible values. A validation error is thrown if another value is specified.
Value	*String*	The value you set or a preconfigured default.
Required	*Boolean*	Whether the property is required to connect.
Category	*String*	The category of the connection property.
IsSessionProperty	*String*	Whether the property is a session property, used to save information about the current connection.

sys_sqlinfo

Describes the SELECT query processing that the adapter can offload to the data source.

Collaborative Query Processing

When working with data sources that do not support SQL-92, you can query the sys_sqlinfo view to determine the query capabilities of the underlying APIs, expressed in SQL syntax. See SQL Compliance for SQL syntax details.

Discovering the Data Source's SELECT Capabilities

Below is an example data set of SQL capabilities. Some aspects of SELECT functionality are returned in a comma-separated list if supported; otherwise, the column contains NO.

Name	**Possible Values**
AGGREGATE_FUNCTIONS	AVG, COUNT, MAX, MIN, SUM, DISTINCT
COUNT	YES
SUPPORTED_OPERATORS	=, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR
GROUP_BY	NO_RELATION
OUTER_JOINS	YES
OJ_CAPABILITIES	NESTED, LEFT, RIGHT, INNER, NOT_ORDERED, ALL_COMPARISON_OPS
SUBQUERIES	COMPARISON, EXISTS, IN, CORRELATED_SUBQUERIES, QUANTIFIED
STRING_FUNCTIONS	ASCII,CHAR,CONCAT,LEFT,LTRIM,REPLACE,RIGHT,RTRIM,SOUNDEX,SPACE,SUBSTRING
NUMERIC_FUNCTIONS	ABS,ACOS,ASIN,ATAN,CEILING,COS,COT,DEGREES,EXP,FLOOR,LOG,LOG10,PI,POWER,RADIANS,RAND,ROUND,SIGN,SIN,SQRT,TAN
TIMEDATE_FUNCTIONS	CURRENT_DATE,CURRENT_TIMESTAMP,MONTH,YEAR
IDENTIFIER_QUOTE_OPEN_CHAR	[
IDENTIFIER_QUOTE_CLOSE_CHAR	]

The following query retrieves the operators that can be used in the WHERE clause:

```
SELECT * FROM sys_sqlinfo WHERE Name='SUPPORTED_OPERATORS'
```

Note that individual tables may have different limitations or requirements on the WHERE clause; refer to the NoSQL Database section for more information.

Columns

Name	Type	Description
NAME	*String*	A component of SQL syntax, or a capability that can be processed on the server.
VALUE	*String*	Detail on the supported SQL or SQL syntax.

Describes the SELECT query processing that the adapter can offload to the data source.

Collaborative Query Processing

When working with data sources that do not support SQL-92, you can query the sys_sqlinfo view to determine the query capabilities of the underlying APIs, expressed in SQL syntax. See SQL Compliance for SQL syntax details.

Discovering the Data Source's SELECT Capabilities

Below is an example data set of SQL capabilities. Some aspects of SELECT functionality are returned in a comma-separated list if supported; otherwise, the column contains NO.

Name	**Possible Values**
AGGREGATE_FUNCTIONS	AVG, COUNT, MAX, MIN, SUM, DISTINCT
COUNT	YES
SUPPORTED_OPERATORS	=, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR
GROUP_BY	NO_RELATION
OUTER_JOINS	YES
OJ_CAPABILITIES	NESTED, LEFT, RIGHT, INNER, NOT_ORDERED, ALL_COMPARISON_OPS

SUBQUERIES	COMPARISON, EXISTS, IN, CORRELATED_SUBQUERIES, QUANTIFIED
STRING_FUNCTIONS	ASCII,CHAR,CONCAT,LEFT,LTRIM,REPLACE,RIGHT,RTRIM,SOUNDEX,SPACE,SUBSTRING
NUMERIC_FUNCTIONS	ABS,ACOS,ASIN,ATAN,CEILING,COS,COT,DEGREES,EXP,FLOOR,LOG,LOG10,PI,POWER,RADIANS,RAND,ROUND,SIGN,SIN,SQRT,TAN
TIMEDATE_FUNCTIONS	CURRENT_DATE,CURRENT_TIMESTAMP,MONTH,YEAR
IDENTIFIER_QUOTE_OPEN_CHAR	[
IDENTIFIER_QUOTE_CLOSE_CHAR	]

The following query retrieves the operators that can be used in the WHERE clause:

```
SELECT * FROM sys_sqlinfo WHERE Name='SUPPORTED_OPERATORS'
```

Note that individual tables may have different limitations or requirements on the WHERE clause; refer to the NoSQL Database section for more information.

Columns

Name	Type	Description
NAME	*String*	A component of SQL syntax, or a capability that can be processed on the server.
VALUE	*String*	Detail on the supported SQL or SQL syntax.

sys_identity

Returns information about attempted modifications. The following query retrieves the Ids of the modified rows in a batch operation.

```
SELECT * FROM sys_identity
```

Columns

Name	Type	Description
Id	*String*	The database-generated Id returned from a data modification operation.
Batch	*String*	An identifier for the batch. 1 for a single operation.
Operation	*String*	The result of the operation in the batch: INSERTED, UPDATED, or DELETED.
Message	*String*	SUCCESS or an error message if the update in the batch failed.

Returns information about attempted modifications. The following query retrieves the Ids of the modified rows in a batch operation.

```
SELECT * FROM sys_identity
```

Columns

Name	Type	Description
Id	*String*	The database-generated Id returned from a data modification operation.
Batch	*String*	An identifier for the batch. 1 for a single operation.
Operation	*String*	The result of the operation in the batch: INSERTED, UPDATED, or DELETED.
Message	*String*	SUCCESS or an error message if the update in the batch failed.

SQL Compliance

The Amazon DynamoDB Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See NoSQL Database for information on the capabilities of the Amazon DynamoDB API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Amazon DynamoDB adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | > | < | >= | <= | IS NULL | IS NOT
NULL | BEGINSWITH | CONTAINS | NOT CONTAINS | BETWEEN | IN | NOT IN
| LIKE | NOT LIKE | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Account
```

Rename a column:

```
SELECT "Name" AS MY_Name FROM Account
```

Search data:

```
SELECT * FROM Account WHERE FirstName <> 'Bob';
```

The Amazon DynamoDB APIs support the following operators in the WHERE clause: =, !=, >, <, >=, <=, IS NULL, IS NOT NULL, BEGINSWITH, CONTAINS, NOT CONTAINS, BETWEEN, IN, NOT IN, LIKE, NOT LIKE, AND, OR.

```
SELECT * FROM Account WHERE FirstName <> 'Bob';
```

Projection Functions

JSON_AVG(json, jsonpath)

Computes the average value of a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_COUNT(json, jsonpath)

Returns the number of elements in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MAX(json, jsonpath)

Gets the maximum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MIN(json, jsonpath)

Gets the minimum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_SUM(json, jsonpath)

Computes the sum of the elements in a JSON within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_EXTRACT(json, jsonpath)

Selects any value in a JSON array or object. The path to the array is specified in the jsonpath argument. Return value is numeric or null.

json: The JSON document to extract.

jsonpath: The XPath used to select the nodes. The JSONPath must be a string constant. The values of the nodes selected will be returned in a token-separated list.

XML_EXTRACT(xml, xpath [, separator])

Extracts an XML document using the specified XPath to flatten the XML. A comma is used to separate the outputs by default, but this can be changed by specifying the third parameter.

xml: The XML document to extract.

xpath: The XPath used to select the nodes. The nodes selected will be returned in a token-separated list.

separator: The optional token used to separate the items in the flattened response. If this is not specified, the separator will be a comma.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Name INTO "csv://Account.txt" FROM "Account" WHERE
FirstName = 'Bob'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Name INTO "csv://Account.txt;delimiter=tab" FROM
"Account" WHERE FirstName = 'Bob'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Account SET Name='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:dynamodb:Access Key=xxx;Secret
Key=xxx;Domain=amazonaws.com;Region=OREGON;",);
String cmd = "DELETE FROM Account WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The adapter allows you to access data in Amazon DynamoDB using a standard database-like interface. Amazon DynamoDB is a highly scalable NoSQL cloud database that is very different from a regular database. In this section we describe how we model schemaless Amazon DynamoDB tables as regular Tables and Stored Procedures.

The adapter can dynamically detect schemas at connection time. See Automatic Schema Discovery for more information on defining schemas implicitly at connection time. This method is useful if the structure of your data is volatile.

You can also persist schemas in static schema definitions. The adapter's schema files have a simple format. See Custom Schema Definitions for more information on defining and extending static schemas.

A full list is available upon customer request.

TDV Email Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

The User and Password properties, under the Authentication section, must be set to valid credentials. The Server must be specified to retrieve emails and the SMTPServer must be specified to send emails.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Email Service	Optimizes the IMAP connection for the service you are working with.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Include Message	Whether to include message body content and attachment data or not.

Is HTML	Determines whether to return the MesssageBody as HTML or plain-text.
Keep Alive	Determines whether to keep the connection alive across instances.
List Mailboxes	Whether to list all mailboxes or just the subscribed IMAP mailboxes. IMAP Only.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Max Items	Maximum number of items to return.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password of the email account used to authenticate.
Port	The port of the mail server. The default value is 143 for IMAP and 110 for POP.
Protocol	The type of email server to connect to.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to IMAP from the provider.
Server	The name or address of the mail server.

SMTP Port	The server port for SMTP (default 25).
SMTP Server	The name or address of the mail server (SMTP server).
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UID Mode	If true, permanent message Ids are used instead of the default temporary Ids.
User	The user of the Email account used to authenticate.

Email Service

Optimizes the IMAP connection for the service you are working with.

Data Type

string

Default Value

"Other"

Remarks

Possible values include AOL, Gmail, Outlook, Yahoo, and Other.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Email and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Include Message

Whether to include message body content and attachment data or not.

Data Type

bool

Default Value

false

Remarks

Whether to include message body content and attachment data or not.

Is HTML

Determines whether to return the MesssageBody as HTML or plain-text.

Data Type

bool

Default Value

true

Remarks

Determines whether to return the MesssageBody as HTML or plain-text.

Keep Alive

Determines whether to keep the connection alive across instances.

Data Type

bool

Default Value

true

Remarks

Determines whether to keep the connection alive across instances.

List Mailboxes

Whether to list all mailboxes or just the subscribed IMAP mailboxes. IMAP Only.

Data Type

string

Default Value

"All"

Remarks

Whether to list all mailboxes or just the subscribed IMAP mailboxes. IMAP Only.

All

Subscribed

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Max Items

Maximum number of items to return.

Data Type

string

Default Value

"100"

Remarks

Maximum number of items to return.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password of the email account used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with User, this field is used to authenticate to the email servers.

Port

The port of the mail server. The default value is 143 for IMAP and 110 for POP.

Data Type

string

Default Value

""

Remarks

A valid port number (a value between 1 and 65535) is required for the connection to take place. The property must be set before a connection is attempted and cannot be changed once a connection is established.

Protocol

The type of email server to connect to.

Data Type

string

Default Value

"IMAP"

Remarks

Possible values include POP and IMAP. The default value of this field is IMAP.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to IMAP from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Server

The name or address of the mail server.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address or the domain name of the mail server. It must be set before a connection is attempted and cannot be changed once a connection is in progress.

If this property is set to a domain name, a DNS request is initiated.

If the connection is configured to use a SOCKS firewall, the value assigned to this property may be preceded with a "*". If this is the case, the host name is passed to the firewall unresolved and the firewall performs the DNS resolution.

SMTP Port

The server port for SMTP (default 25).

Data Type

string

Default Value

"25"

Remarks

A valid port number (a value between 1 and 65535) is required for the connection to take place. The property must be set before a connection is attempted and cannot be changed once a connection is established.

SMTP Server

The name or address of the mail server (SMTP server).

Data Type

string

Default Value

""

Remarks

This property specifies the IP address or domain name of the mail server. It must be set before a connection is attempted and cannot be changed once a connection is in progress.

If this property is set to a domain name, a DNS request is initiated.

If the connection is configured to use a SOCKS firewall, the value assigned to this property may be preceded with a "*". If this is the case, the host name is passed to the firewall unresolved and the firewall performs the DNS resolution.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example

A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

UID Mode

If true, permanent message Ids are used instead of the default temporary Ids.

Data Type

bool

Default Value

false

Remarks

The default value for UIDMode is false.

User

The user of the Email account used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with Password, this field is used to authenticate to the email servers.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Email Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

SQL Compliance

The Email Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Email API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Email adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | AND |
OR | BETWEEN | IN } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM [INBOX]
```

Rename a column:

```
SELECT "Subject" AS MY_Subject FROM [INBOX]
```

Search data:

```
SELECT * FROM [INBOX] WHERE Subject = 'Test';
```

The Email APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, AND, OR, BETWEEN, IN.

```
SELECT * FROM [INBOX] WHERE Subject = 'Test';
```

Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM [INBOX]
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT From, Subject INTO "csv://[INBOX].txt" FROM "[INBOX]" WHERE
Subject = 'Test'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT From, Subject INTO "csv://[INBOX].txt;delimiter=tab" FROM
"[INBOX]" WHERE Subject = 'Test'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO [INBOX] (Subject) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "Spam");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE [INBOX] SET Subject='Spam' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:email:Port=993;Server=imap.gmail
.com;Password=password;User=user;Protocol=IMAP;SMTP Port=587;SMTP
Server=smtp.gmail.com;",);
String cmd = "DELETE FROM [INBOX] WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Supported Mail Protocols

The Email Adapter connects to mail servers using standard mail protocols for IMAP, POP, and SMTP, as defined by the following RFCs:

For IMAP, the adapter implements a standard IMAP client, as specified in RFC 1730 and RFC 2060.

For POP, the adapter implements a standard Internet post office (POP3) client, as specified in RFC 1725.

And, finally, for SMTP, the adapter implements a standard SMTP client, as specified in RFC 821.

IMAP

IMAP Specific Information

Email address fields accept the following formats for email addresses:

"Friendly Name" <address@company.com>

address@company.com

SELECT

Email will be listed newest to oldest. By default, the MessageBody and Headers will not be returned when listing multiple messages.

Note: By default, the max number of messages returned will be 100. This can be changed by setting either LIMIT or MaxItems. If you wish to return all email within a mailbox, you will need to specify a value of 0 or -1.

List Mail Messages

To list messages within a specific mailbox, specify the mailbox as the table. This will list the most recent messages.

Alternatively, you can specify a range of message Ids to list. This includes the following formats:

```
WHERE Id='10' //For returning only the message with an Id of 10.
```

```
WHERE Id BETWEEN 10 AND 20 //For returning all messages from 10 to
20.
WHERE Id IN (1,3,5) //For returning messages 1, 3, 5.
```

Search Email

To search mail, simply specify a value for any of the following columns in the WHERE clause: To, From, BCC, CC, Subject, MessageBody, Flags, Labels, Size, or Date. This includes complex statements. For example:

```
SELECT * FROM [Inbox] WHERE MessageBody LIKE "Test" AND ([From] =
test1@email.com OR [From] = test2@email.com) AND Date>'1-1-2012'
```

This will return all messages where the message body contains the text 'Test', and the message is from either test1@email.com or test2@email.com.

In addition to searching by standard columns, you can also search by attachment file names and by whether or not an attachment is on the email. For example:

```
SELECT * FROM [Inbox] WHERE HasAttachment='True' AND
Attachments='.txt'
```

Use of parentheses is also supported for complex WHERE clauses.

As an alternative to the above method of creating search criteria, the SearchCriteria pseudo column is also available. This will accept any valid IMAP search criteria as specified by RFC.

INSERT

Please refer to the SMTP section.

UPDATE

To move an email from one mailbox to another, you will need to specify the Mailbox in the SET clauses of the message and the ID in the WHERE clause. For example:

```
UPDATE [Inbox] SET Mailbox='NewMailboxName' WHERE Id='MessageId'
```

Id may consist of a single message number, a range of messages specified by two message numbers separated by ':' (e.g. "1:5"), and/or individual message numbers separated by ',' (e.g. "1:5,7,10").

When moving an email from one mailbox to another, you may not specify any additional updates.

Additional Notes

By default, the number of messages returned per page will be 25. To change this, you can set the ItemsPerPage pseudo column.

All message Ids returned are temporary Ids and may change in subsequent requests to the server. To use static Ids, you can set UIDMode=true. However, only listing messages is supported with UIDs.

POP

POP Specific Information

Email address fields contained within this table accept the following formats:

'Friendly Name' <address@company.com>

address@company.com

SELECT

Email will be listed newest to oldest. By default, the MessageBody and Headers will not be returned when listing multiple messages.

Note: By default, the max number of emails returned will be 100. This can be changed by setting either LIMIT or MaxItems. If you wish to return all emails, you will need to specify a value of 0 or -1.

List Emails

To list messages, simply call SELECT after specifying the account information. This will list the most recent messages first.

To specify a single Id, which will return headers and the message body, you can specify the following:

```
WHERE Id='10' //For returning only the message with an Id of 10.
```

To list messages beginning with a specific Id, you can set the StartId property in the WHERE clause. This will cause messages to begin with the Id you specified, to the oldest message available within the LIMIT.

```
WHERE StartId='45' //Returns messages 45 through 1
```

Search Emails

Searching is not supported by the POP protocol. In order to search emails, you must use IMAP.

INSERT

Please refer to the SMTP section.

UPDATE

Updating is not supported by the POP protocol.

Additional notes

By default, the number of messages returned per page will be 25. To change this, you can set the ItemsPerPage pseudo column.

All message Ids returned are temporary Ids and may change in subsequent requests to the server. To use static Ids, you can set UIDMode=true. However, only listing messages is supported with UIDs.

SMTP

SMTP Specific Information

Email address fields accept both the email addresses and the email address accompanied by the username; for example, "Friendly Name" <address@company.com>

INSERT

When sending mail with either POP or IMAP, SMTP is used to send the message to the mail server.

To send mail, you can insert a row into the table. Required fields are Subject, To, and MessageBody. For example:

```
INSERT INTO [Inbox] (Subject, MessageBody, To) VALUES ('Test
Subject','Body Text','address@company.com')
```

Additional notes

By default, the From field will be populated with the email address supplied in the account. If it is not a complete email address, be sure to set From before sending a message.

Data Model

The Email Adapter models Email entities as relational tables and stored

procedures.

A full list is available upon customer request.

TDV Elasticsearch Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to Elasticsearch

Set the Server and Port connection properties, in addition to the authentication properties. The Elasticsearch Adapter uses X-Pack Security for authentication and TLS/SSL encryption.

Authenticating to Elasticsearch

Set the User and Password properties and/or use PKI (public key infrastructure) to authenticate. Once the adapter is connected, X-Pack will then perform user authentication and grant role permissions based on the realms you have configured.

To use PKI, set the SSLClientCert, SSLClientCertType, SSLClientCertSubject, and SSLClientCertPassword properties.

Note: TLS/SSL and client authentication must be enabled on X-Pack to use PKI.

Securing Elasticsearch Connections

To enable TLS/SSL in the adapter, prefix the Server value with 'https://'.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Flatten Arrays	Set FlattenArrays to the number of nested array elements you want to return as table columns. By default, nested arrays are returned as strings of JSON.
Flatten Objects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
Generate Schema Files	Indicates the user preference as to when schemas should be generated and saved.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Max Results	The maximum number of total results to return from Elasticsearch when using the default Search API.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per request from Elasticsearch.
Password	The password used to authenticate to Elasticsearch.
Port	The port for the Elasticsearch REST server.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Query Passthrough	This option allows you to pass queries to Elasticsearch using Elasticsearch's Search DSL language, which includes Query DSL.
Readonly	You can use this property to enforce read-only access to Elasticsearch from the provider.
Row Scan Depth	The maximum number of rows to scan when generating table metadata. Set this property to gain more control over how the provider detects arrays.
Scroll Duration	Specifies the time unit to use when retrieving results via the Scroll API.
Server	The host name or IP address of the Elasticsearch REST server.
SSL Client Cert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSL Client Cert Password	The password for the TLS/SSL client certificate.
SSL Client Cert Subject	The subject of the TLS/SSL client certificate.
SSL Client Cert Type	The type of key store containing the TLS/SSL client certificate.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
User	The user who is authenticating to Elasticsearch.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Elasticsearch and traffic flows back and forth through the proxy.

SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Flatten Arrays

Set FlattenArrays to the number of nested array elements you want to return as table columns. By default, nested arrays are returned as strings of JSON.

Data Type

string

Default Value

""

Remarks

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

Set FlattenArrays to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

For example, you can return an arbitrary number of elements from an array of strings:

```
"employees": [
  {
    "name": "John Smith",
    "age": 34
  },
  {
    "name": "Peter Brown",
    "age": 26
  },
  {
    "name": "Paul Jacobs",
    "age": 30
  }
]
```

When FlattenArrays is set to 2, the preceding array is flattened into the following table:

Column Name	Column Value
employees.0.name	John Smith
employees.0.age	34

employees.1.name	Peter Brown
employees.1.age	26

See JSON Functions to use JSON paths to work with unbounded arrays.

Flatten Objects

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Data Type

bool

Default Value

true

Remarks

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON. The property name is concatenated onto the object name with a period to generate the column name.

For example, you can flatten the nested objects below at connection time:

```
"manager": {
  "name": "Alice White",
  "age": 30
}
```

When FlattenObjects is set to true, the preceding object is flattened into the following table:

Column Name	Column Value
manager.name	Alice White
manager.age	30

Generate Schema Files

Indicates the user preference as to when schemas should be generated and saved.

Data Type

string

Default Value

"Never"

Remarks

GenerateSchemaFiles enables you to persist a relational view of Elasticsearch types (tables) or queries; this property outputs schemas to .rsd files in the path specified by Location.

See the following sections for ways to generate schema files or for alternatives. Schema files are easy to edit, but they are static: If you want to regenerate a file, you will first need to delete it.

Available settings are the following:

Never: A schema file will never be generated.

OnUse: A schema file will be generated the first time a table is referenced, provided the schema file for the table does not already exist.

OnStart: A schema file will be generated at connection time for any tables that do not currently have a schema file.

Generate Schemas with SQL

By setting GenerateSchemaFiles to OnUse, you can create views on queries you execute. For example, you could specify the XPath to a column value in your SELECT query with an SQL function -- see JSON Functions for examples. See Query Mapping for more information on how to write SQL queries to Elasticsearch.

If you want to write a new query for the view, you will need to first delete the old schema from the Location folder.

Generate Schemas on Connection

Another way to use this property is to obtain schemas for every table in your database when you connect. To do so, set GenerateSchemaFiles to OnStart and connect.

See Automatic Schema Discovery to use connection properties to fine-tune the tables rendered.

Editing Schemas

Schema files have a simple format that makes them easy to modify. See Custom Schema Definitions for an end-to-end guide.

Alternatives to Static Schemas

If your data structures are volatile, consider setting GenerateSchemaFiles to Never and using dynamic schemas. See Automatic Schema Discovery for more information about dynamic schemas.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Max Results

The maximum number of total results to return from Elasticsearch when using the default Search API.

Data Type

string

Default Value

"10000"

Remarks

This property corresponds to the Elasticsearch *index.max_result_window* index setting. Thus the default value is 10000, which is Elasticsearch's default limit.

This value is not applicable when using the Scroll API. Set ScrollDuration to use this API.

When a LIMIT is specified in a query, the LIMIT will be taken into account provided it is less than MaxResults. Otherwise the number of results returned will be limited to the MaxResults value.

If you receive an error stating that the result window is too large, this is caused by the MaxResults value being greater than the Elasticsearch *index.max_result_window* index setting. You can either change the MaxResults value to match the *index.max_result_window* index setting or use the Scroll API by setting ScrollDuration.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per request from Elasticsearch.

Data Type

string

Default Value

"10000"

Remarks

The PageSize can control the number of results received per request from Elasticsearch on a given query.

The default value is 10000, which is Elasticsearch's default limit (based on the Elasticsearch *index.max_result_window* index setting).

Password

The password used to authenticate to Elasticsearch.

Data Type

string

Default Value

""

Remarks

The password used to authenticate to Elasticsearch.

Port

The port for the Elasticsearch REST server.

Data Type

string

Default Value

"9200"

Remarks

The port the Elasticsearch REST server is bound to.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Query Passthrough

This option allows you to pass queries to Elasticsearch using Elasticsearch's Search DSL language, which includes Query DSL.

Data Type

bool

Default Value

false

Remarks

Setting this property to True enables the adapter to pass an Elasticsearch query as-is to Elasticsearch. The supported query syntax is JSON using the query passthrough syntax described below.

The JSON Passthrough Query Syntax supports the following elements:

Element Name	Function
index	The Elasticsearch index (or schema) to query. This is a JSON element that takes a string value.
type	The Elasticsearch type (or table) to query within *index*. This is a JSON element that takes a string value.
docid	The Id of the document to query within *index.type*. This is a JSON element that takes a string value.
apiendpoint	The Elasticsearch API Endpoint to query. Default value is '_search'. This is a JSON element that takes a string value.
requestdata	The raw Elasticsearch Search DSL that will be sent to Elasticsearch as is. The value is a JSON object that maps directly to the format required by Elasticsearch.

The *index*, *type*, *docid*, and *apiendpoint* are used to generate the URL where the *requestdata* will be sent. The URL is generated using the following format: [Server]:[Port]/[index]/[type]/[docid]/[apiendpoint]. If any of the JSON passthrough elements are not specified, they will not be added to the URL.

Below is an example of a passthrough query. This example will retrieve the first 10 documents from *megacorp.employee* that contain a *last_name* of 'smith'. The results will be ordered by *first_name* in descending order.

```
{
  "index": "megacorp",
  "type": "employee",
  "requestdata":
  {
    "from": 0,
    "size": 10,
    "query": {"bool":{"must":{"term":{"last_name":"smith"}}}},
    "sort": {"first_name":{"order":"desc"}}
  }
}
```

When using QueryPassthrough queries, the metadata is determined by the data returned in the response. RowScanDepth identifies the depth of the records that will be scanned to determine the metadata (columns and types). Since the metadata is based on the response data, passthrough queries may display different metadata than a similar query performed using the SQL syntax (where the metadata is retrieved directly from Elasticsearch).

Readonly

You can use this property to enforce read-only access to Elasticsearch from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Row Scan Depth

The maximum number of rows to scan when generating table metadata. Set this property to gain more control over how the provider detects arrays.

Data Type

string

Default Value

"100"

Remarks

This property is used when generating table metadata and specifically is used to identify arrays within the data. Elasticsearch allows any field to be an array and does not identify which fields are arrays in the mapping data. Thus RowScanDepth rows will be queried and scanned to identify if any of the fields contain arrays.

When QueryPassthrough is set to True, the columns in a table must be determined by scanning the data returned in the request. This value determines the maximum number of rows that will be scanned to determine the table metadata. The default value is 100.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

Scroll Duration

Specifies the time unit to use when retrieving results via the Scroll API.

Data Type

string

Default Value

"1m"

Remarks

When a nonzero value is specified, the Scroll API will be used.

The time unit specified will be sent in each request made to Elasticsearch to specify how long the server should keep the search context alive. The value specified only needs to be long enough to process the previous batch of results (not to process all the data). This is because the ScrollDuration value will be sent in each request, which will extend the context time.

Once all the results have been retrieved, the search context will be cleared.

The format for this value is: [integer][time unit]. For example: 1m = 1 minute.

Setting this property to '0' will cause the default Search API to be used. In such a case, the maximum number of results that can be returned are equal to MaxResults.

Supported Time Units:

Value	Description
y	Year
M	Month
w	Week
d	Day
h	Hour
m	Minute
s	Second
ms	Milli-second

Server

The host name or IP address of the Elasticsearch REST server.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the Elasticsearch REST server.

To use SSL, prefix the host name or IP address with 'https://' and set SSL connection properties such as SSLServerCert.

SSL Client Cert

The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The SSLClientCertType field specifies the type of the certificate store specified by SSLClientCert. If the store is password protected, specify the password in SSLClientCertPassword.

SSLClientCert is used in conjunction with the SSLClientCertSubject field in order to specify client certificates. If SSLClientCert has a value, and SSLClientCertSubject is set, a search for a certificate is initiated. Please refer to the SSLClientCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

SSL Client Cert Password

The password for the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

SSL Client Cert Subject

The subject of the TLS/SSL client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

SSL Client Cert Type

The type of key store containing the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.

XMLBLOB	The certificate store is a string that contains a certificate in XML format.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801c bb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

User

The user who is authenticating to Elasticsearch.

Data Type

string

Default Value

""

Remarks

The user who is authenticating to Elasticsearch.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Elasticsearch Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Accessing NoSQL Tables

The adapter implements Automatic Schema Discovery that is highly configurable. The following sections outline the adapter's defaults and link to ways to further customize.

Flattening Nested JSON

By default, the adapter projects columns over the properties of objects, including objects nested in objects. Arrays are returned as JSON strings, by default. You can use the following properties to access array elements, including objects nested in arrays.

FlattenArrays: Set this property to the number of array elements that you want to return as column values. You can also use this property with FlattenObjects to extract the properties of objects nested in arrays.

FlattenObjects: By default, this is true; that is, the properties of objects and nested objects are returned as columns. When you set FlattenArrays, objects nested in the specified array elements are also flattened and returned as columns.

Other mechanisms for accessing nested objects are detailed in Searching with SQL.

Fine Tuning Data Access

You can use the following properties to gain greater control over Elasticsearch API features and the strategies the adapter uses to surface them:

GenerateSchemaFiles: This property enables you to persist table metadata in static schema files that are easy to customize, to persist your changes to column data types, for example. You can set this property to "OnStart" to generate schema files for all tables in your database at connection. The resulting schemas are based on the connection properties you use to configure Automatic Schema Discovery

.

Or, you can set this property to "OnUse" to generate schemas based on a query.

To use the resulting schema files, set the Location property to the folder containing the schemas.

QueryPassthrough: This property enables you to use Elasticsearch's Search DSL language instead of SQL.

RowScanDepth: This property determines the number of rows that will be scanned to detect column data types when generating table metadata. This property applies if you are working with the dynamic schemas generated from Automatic Schema Discovery

or if you are using QueryPassthrough.

Fine Tuning Performance

PageSize: This property enables you to optimize performance based on your resource provisioning. Paging has an impact on sorting performance in a distributed system, as each shard must first sort results before submitting them to the coordinating server.

By default, the adapter requests a page size of 10,000. This is the default *index.max_result_window* setting in Elasticsearch.

MaxResults: This property sets a limit on the results for queries at connection time, without requiring that you specify a LIMIT clause. By default, this is the same value as the *index.max_result_window* setting in Elasticsearch.

If you are using the Scroll API, set ScrollDuration instead.

ScrollDuration: This property specifies how long the server should keep the search context alive. Setting this property to a nonzero value and time unit enables the Scroll API.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the connection properties needed to authenticate and connect. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

Searching with SQL

Elasticsearch is a document-oriented database that provides high performance searching, flexibility, and scalability. These features are not necessarily incompatible with a standards-compliant query language like SQL-92. In this section we will show various schemes that the adapter offers to bridge the gap with relational SQL and an Elasticsearch database.

The adapter models Elasticsearch objects into relational tables and translates SQL queries into Elasticsearch queries to get the requested data. See Schema Mapping for more details on how Elasticsearch objects are mapped to tables to generate schemas. See Query Mapping for more details on how various Elasticsearch operations are represented as SQL.

The Automatic Schema Discovery scheme automatically finds the data types by retrieving the mapping for the Elasticsearch type. You can use RowScanDepth, FlattenArrays, and FlattenObjects to control the relational representation of the collections in Elasticsearch.

Optionally, you can use Custom Schema Definitions to project your chosen relational structure on top of a Elasticsearch object. This allows you choose your own column names, their data types, and the location of their values in the collection.

When GenerateSchemaFiles is set, you can persist schemas for all collections in the database or for the results of SELECT queries.

Schema Mapping

The Elasticsearch Adapter models the Elasticsearch REST APIs as relational tables and stored procedures that can be accessed with standard SQL. This enables access from standards-based tools.

The table definitions are dynamically retrieved. When you connect, the adapter connects to Elasticsearch and retrieves the schemas, list of tables, and the metadata for the tables by querying the Elasticsearch REST server. Any changes to the remote data are immediately reflected in your queries.

The following table maps Elasticsearch concepts to relational ones:

Elasticsearch Concept	SQL Concept
Index	Schema
Type	Table
Alias	View
Document	Row (each document is a row and the document's JSON structure is represented as columns)
Field	Column

Automatic Schema Discovery

The adapter automatically infers a relational schema by retrieving the mapping of the Elasticsearch type. The columns and data types are generated from the retrieved mapping.

Detecting Arrays

Any field within Elasticsearch can be an array of values, but this is not explicitly defined within the mapping. To account for this, the adapter will query the data to detect if any fields contain arrays. The number of Elasticsearch documents retrieved during this array scanning is based on the RowScanDepth property.

Elasticsearch nested types are special types that denote an array of objects and thus will always be treated as such when generating the metadata.

Detecting Columns

The columns identified during the discovery process depend on the FlattenArrays and FlattenObjects properties.

Example Data Set

To provide an example of how these options work, consider the following mapping (where 'insured' is the name of the table):

```
{
  "insured": {
    "properties": {
      "name": { "type":"string" },
      "address": {
        "street": { "type":"string" },
        "city": { "type":"string" },
        "state": { "type":"string" }
      },
      "insured_ages": { "type": "integer" },
      "vehicles": {
        "type": "nested",
        "properties": {
          "year": { "type":"integer" },
          "make": { "type":"string" },
          "model": { "type":"string" },
          "body_style" { "type": "string" }
        }
      }
    }
  }
```

```
}
```

Also consider the following example data for the above mapping:

```
{
  "_source": {
    "name": "John Smith",
    "address": {
      "street": "Main Street",
      "city": "Chapel Hill",
      "state": "NC"
    },
    "insured_ages": [ 17, 43, 45 ],
    "vehicles": [
      {
        "year": 2015,
        "make": "Dodge",
        "model": "RAM 1500",
        "body_style": "TK"
      },
      {
        "year": 2015,
        "make": "Suzuki",
        "model": "V-Strom 650 XT",
        "body_style": "MC"
      },
      {
        "year": 2012,
        "make": "Honda",
        "model": "Accord",
        "body_style": "4D"
      }
    ]
  }
}
```

Using FlattenObjects

If FlattenObjects is set, all nested objects will be flattened into a series of columns. The above example will be represented by the following columns:

Column Name	Data Type	Example Value
name	String	John Smith
address.street	String	Main Street
address.city	String	Chapel Hill
address.state	String	NC

insured_ages	String	[17, 43, 45]
vehicles	String	[{ "year": "2015", "make": "Dodge", ... }, { "year": "2015", "make": "Suzuki", ... }, { "year": "2012", "make": "Honda", ... }]

If FlattenObjects is not set, then the address.street, address.city, and address.state columns will not be broken apart. The address column of type string will instead represent the entire object. Its value would be the following:

```
{street: "Main Street", city: "Chapel Hill", state: "NC"}
```

See JSON Functions for more details on working with JSON aggregates.

Using FlattenArrays

The FlattenArrays property can be used to flatten array values into columns of their own. This is only recommended for arrays that are expected to be short. It is best to leave unbounded arrays as they are and piece out the data for them as needed using JSON Functions.

Note: Only the top-most array will be flattened. Any subarrays will be represented as the entire array.

The FlattenArrays property can be set to 3 to represent the arrays in the example above as follows (this example is with FlattenObjects not set):

Column Name	Data Type	Example Value
insured_ages	String	[17, 43, 45]
insured_ages.0	Integer	17
insured_ages.1	Integer	43
insured_ages.2	Integer	45
vehicles	String	[{ "year": "2015", "make": "Dodge", ... }, { "year": "2015", "make": "Suzuki", ... }, { "year": "2012", "make": "Honda", ... }]
vehicles.0	String	{ "year": "2015", "make": "Dodge", "model": "RAM 1500", "body_style": "TK" }

vehicles.1	String	{ "year": "2015", "make": "Suzuki", "model": "V-Strom 650 XT", "body_style": "MC" }
vehicles.2	String	{ "year": "2012", "make": "Honda", "model": "Accord", "body_style": "4D" }

Using Both FlattenObjects and FlattenArrays

If FlattenObjects is set along with FlattenArrays (set to 1 for brevity), the vehicles field will be represented as follows:

Column Name	**Data Type**	Example Value
vehicles	String	[{ "year": "2015", "make": "Dodge", ... }, { "year": "2015", "make": "Suzuki", ... }, { "year": "2012", "make": "Honda", ... }]
vehicles.0.year	String	2015
vehicles.0.make	String	Dodge
vehicles.0.model	String	RAM 1500
vehicles.0.body_style	String	TK

JSON Functions

The adapter can return JSON structures as column values. The adapter enables you to use standard SQL functions to work with these JSON structures. The examples in this section use the following array:

```
[
     { "grade": "A", "score": 2 },
     { "grade": "A", "score": 6 },
     { "grade": "A", "score": 10 },
     { "grade": "A", "score": 9 },
     { "grade": "B", "score": 14 }
]
```

JSON_EXTRACT

The JSON_EXTRACT function can extract individual values from a JSON object. The following query returns the values shown below based on the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_EXTRACT(grades,'[0].grade') AS Grade,
JSON_EXTRACT(grades,'[0].score') AS Score FROM Students;
```

Column Name	Example Value
Grade	A
Score	2

JSON_COUNT

The JSON_COUNT function returns the number of elements in a JSON array within a JSON object. The following query returns the number of elements specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_COUNT(grades,'[x]') AS NumberOfGrades FROM
Students;
```

Column Name	Example Value
NumberOfGrades	5

JSON_SUM

The JSON_SUM function returns the sum of the numeric values of a JSON array within a JSON object. The following query returns the total of the values specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_SUM(score,'[x].score') AS TotalScore FROM
Students;
```

Column Name	Example Value

TotalScore	41

JSON_MIN

The JSON_MIN function returns the lowest numeric value of a JSON array within a JSON object. The following query returns the minimum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MIN(score,'[x].score') AS LowestScore FROM
Students;
```

Column Name	Example Value
LowestScore	2

JSON_MAX

The JSON_MAX function returns the highest numeric value of a JSON array within a JSON object. The following query returns the maximum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MAX(score,'[x].score') AS HighestScore FROM
Students;
```

Column Name	Example Value
HighestScore	14

DOCUMENT

The DOCUMENT function can be used to retrieve the entire document as a JSON string. See the following query and its result as an example:

```
SELECT DOCUMENT(*) from Employee;
```

The query above will return the entire document as shown.

```
{
  "_index": "megacorp",
```

```
  "_type": "employee",
  "_id": "2",
  "_score": 1,
  "_source": {
    "first_name": "Jane",
    "last_name": "Smith",
    "age": 32,
    "about": "I like to collect rock albums",
    "interests": [
      "music"
    ]
  }
}
```

Query Mapping

This section describes how SQL statements are interpreted and translated into Elasticsearch queries. Examples are also provided to explain the behavior of various queries.

Query/Filter Context and Scoring

When the *_score* column is selected, scoring will be requested by issuing a query context request, which scores the quality of the search results. By default, results are returned in descending order based on the calculated *_score*. An ORDER BY clause can be specified to change the order of the returned results.

When the *_score* column is not selected, a filter context will be sent, in which case Elasticsearch will not compute scores. The results for these queries will be returned in arbitrary order unless an ORDER BY clause is explicitly specified.

Text Matching and Search

Analyzed fields in Elasticsearch are stored in an inverted index after they are run through an analyzer. Analyzers are customizable and thus can perform a variety of different filters on the data prior to storing them in the inverted index. For example, the default Elasticsearch analyzer will lowercase all the terms.

To demonstrate this point, an analyzed field in Elasticsearch was created with a value of 'Bike'. After being analyzed, the value will be stored in the inverted index (using the default analyzer) as 'bike'. A non-analyzed field, on the other hand, would not analyze the search value and thus would be stored as 'Bike'.

When performing searches, some Elasticsearch query types run the search value through an analyzer (which will make the search case insensitive) and some do

not (making the search case sensitive). Additionally, the default analyzer breaks up fields containing multiple words into separate terms. When performing searches on these fields, Elasticsearch may return records that contain the same words but in a different order. For example, a search is performed using a value of 'blue sky' but a record with 'sky blue' is returned.

To work around these case-sensitivity and ordering issues, the Elasticsearch Adapter will identify the column as analyzed or non-analyzed and will issue the appropriate Elasticsearch query based on the specified operator (such as =) and the search value.

Equals and Not Equals

Where clauses that contain an equals (=) or not equals (!= or <>) filter issue different Elasticsearch queries depending upon the column and data used. Analyzed and non-analyzed columns behave differently and thus different Elasticsearch queries are generated to provide the best search functionality. Additionally, string values generate different query types depending upon whether they contain empty space or not. Below is a breakdown of the rules and behavior for the varying cases.

Analyzed Columns
Analyzed columns are stored after being run through an analyzer. As a result of that, the search values specified will be run through an analyzer on the Elasticsearch server prior to the search. This makes the searches case-insensitive (provided the analyzer used handles casing).

WHERE Clause Examples	Elasticsearch Query Type
WHERE analyzed_column='value'	Query String Query
WHERE analyzed_column='value with spaces'	Match Phrase Query

Non-Analyzed Columns
Non-analyzed columns are stored without being run through an analyzer. Thus, non-analyzed columns are case sensitive and thus search values specified for these columns are case sensitive. If the search value is a single word, the adapter will check the filter with the original casing specified along with three common forms: uppercase, lowercase, and capitalized. If the search value contains multiple words, the search value will be sent as-is and thus is case sensitive.

WHERE Clause Examples	Elasticsearch Query Type
WHERE nonanalyzed_column='myValue'	Query String Query: Four cases are checked - myValue OR MYVALUE OR myvalue OR Myvalue
WHERE nonanalyzed_column='value with spaces'	Wildcard Query

IN and NOT IN

The IN and NOT IN operators function very similarly to the equals and not equals operators.

WHERE Clause Examples	Behavior
WHERE column IN ('value')	Treated as: column='value'
WHERE column NOT IN ('value')	Treated as: column!='value'
WHERE column IN ('value1', 'value2')	Treated as: column='value1' OR column='value2'
WHERE column NOT IN ('value1', 'value2')	Treated as: column!='value1' AND column!='value2'

LIKE and NOT LIKE

The LIKE and NOT LIKE operators allow the use of wildcard characters. The percent sign (%) represents zero, one, or multiple characters. The underscore (_) represents a single character (in which the character must be present).

WHERE Clause Examples	Behavior
WHERE column LIKE 'value'	Treated as: column='value'

WHERE column NOT LIKE 'value'	Treated as: column!='value'
WHERE analyzed_column LIKE 'v_lu%'	Query String Query with wildcards
WHERE nonanalyzed_column LIKE 'v_lu%'	Wildcard Query with wildcards

Custom Schema Definitions

View schemas persist the relational structure the adapter infers for Elasticsearch types and queries. To provide an example of how custom schemas work, we will use the below mapping (where 'insured' is the name of the table).

```
{
  "insured": {
    "properties": {
      "name": { "type":"string" },
      "address": {
        "street": { "type":"string" },
        "city": { "type":"string" },
        "state": { "type":"string" }
      },
      "insured_ages": { "type": "integer" },
      "vehicles": {
        "type": "nested",
        "properties": {
          "year": { "type":"integer" },
          "make": { "type":"string" },
          "model": { "type":"string" },
          "body_style" { "type": "string" }
        }
      }
    }
  }
}
```

Also, consider the following example data for the above mapping:

```
{
  "_source": {
    "name": "John Smith",
    "address": {
      "street": "Main Street",
      "city": "Chapel Hill",
      "state": "NC"
    },
    "insured_ages": [ 17, 43, 45 ],
    "vehicles": [
      {
        "year": 2015,
        "make": "Dodge",
```

```
        "model": "RAM 1500",
        "body_style": "TK"
      },
      {
        "year": 2015,
        "make": "Suzuki",
        "model": "V-Strom 650 XT",
        "body_style": "MC"
      },
      {
        "year": 2012,
        "make": "Honda",
        "model": "Accord",
        "body_style": "4D"
      }
    ]
  }
}
```

Defining a Custom Schema

Schemas persisted when GenerateSchemaFiles is set are placed into the folder specified by the Location property. For example, set GenerateSchemaFiles to "OnUse" and execute a SELECT query:

```
SELECT * FROM insured
```

You can then change column behavior in the resulting schema. The following schema uses the *other:xPath* property to define where the data for a particular column should be retrieved from. Using this model you can flatten arbitrary levels of hierarchy.

The *es_index* and *es_type* attributes specify the Elasticsearch index and type to retrieve. The *es_index* and *es_type* attributes give you the flexibility to use multiple schemas for the same type. If *es_type* is not specified, the filename determines the collection that is parsed.

Below is an example is an example of the column behavior markup. You can find a complete schema in Custom Schema Example.

```
  <rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

    <rsb:info title="StaticInsured" description="Custom Schema for
the Elasticsearch insured data set.">
      <!-- Column definitions -->
      <attr name="_id"                          xs:type="string"
other:xPath="_id"
other:sourceField="_id"                 other:analyzed="true"  />
      <attr name="_score"                       xs:type="double"
other:xPath="_score"
other:sourceField="_score"              other:analyzed="true"  />
      <attr name="name"                         xs:type="string"
other:xPath="_source/name"
other:sourceField="name"                other:analyzed="true"  />
```

```
      <attr name="address.street"              xs:type="string"
other:xPath="_source/address/street"
other:sourceField="address.street"      other:analyzed="true"  />
      <attr name="address.city"                xs:type="string"
other:xPath="_source/address/city"
other:sourceField="address.city"        other:analyzed="true"  />
      <attr name="address.state"               xs:type="string"
other:xPath="_source/address/state"
other:sourceField="address.state"       other:analyzed="true"  />
      <attr name="insured_ages"                xs:type="string"
other:xPath="_source/insured_ages"
other:valueFormat="aggregate" other:sourceField="insured_ages"
other:analyzed="false" />
      <attr name="insured_ages.0"              xs:type="integer"
other:xPath="_source/insured_ages[0]"
other:sourceField="insured_ages"        other:analyzed="false" />
      <attr name="vehicles"                    xs:type="string"
other:xPath="_source/vehicles"
other:valueFormat="aggregate" other:sourceField="vehicles"
other:analyzed="true"  />
      <attr name="vehicles.0.year"             xs:type="integer"
other:xPath="_source/vehicles[0]/year"
other:sourceField="vehicles.year"       other:analyzed="true"  />
      <attr name="vehicles.0.make"             xs:type="string"
other:xPath="_source/vehicles[0]/make"
other:sourceField="vehicles.make"       other:analyzed="true"  />
      <attr name="vehicles.0.model"            xs:type="string"
other:xPath="_source/vehicles[0]/model"
other:sourceField="vehicles.model"      other:analyzed="true"  />
      <attr name="vehicles.0.body_style"       xs:type="string"
other:xPath="_source/vehicles[0]/body_style"
other:sourceField="vehicles.body_style" other:analyzed="true"  />

      <input name="rows@next" desc="Internal attribute used for
paging through data."  />
    </rsb:info>


    <rsb:set attr="es_index" value="auto"/>
    <rsb:set attr="es_type"  value="insured"/>

  </rsb:script>
```

Custom Schema Example

In this section is a complete schema. The info section enables a relational view of an Elasticsearch object. For more details, see Custom Schema Definitions. The table below only supports SELECT commands. INSERT, UPDATE, and DELETE commands are not currently supported.

Use the *es_index* and *es_type* attributes to specify the name of the Elasticsearch type and index you want to retrieve and parse. You can use the *es_index* and *es_type* attributes to define multiple schemas for the same Elasticsearch type.

If *es_type* is not specified, the filename determines the Elasticsearch type that is parsed.

Copy the *rows@next* input as-is into your schema. The operations, such as *elasticsearchadoSelect*, are internal implementations and can also be copied as is.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

  <rsb:info title="StaticInsured" description="Custom Schema for
the Elasticsearch insured data set.">
    <!-- Column definitions -->
    <attr name="_id"                          xs:type="string"
other:xPath="_id"
other:sourceField="_id"                 other:analyzed="true"  />
    <attr name="_score"                       xs:type="double"
other:xPath="_score"
other:sourceField="_score"              other:analyzed="true"  />
    <attr name="name"                         xs:type="string"
other:xPath="_source/name"
other:sourceField="name"                other:analyzed="true"  />
    <attr name="address.street"               xs:type="string"
other:xPath="_source/address/street"
other:sourceField="address.street"      other:analyzed="true"  />
    <attr name="address.city"                 xs:type="string"
other:xPath="_source/address/city"
other:sourceField="address.city"        other:analyzed="true"  />
    <attr name="address.state"                xs:type="string"
other:xPath="_source/address/state"
other:sourceField="address.state"       other:analyzed="true"  />
    <attr name="insured_ages"                 xs:type="string"
other:xPath="_source/insured_ages"
other:valueFormat="aggregate" other:sourceField="insured_ages"
other:analyzed="false" />
    <attr name="insured_ages.0"               xs:type="integer"
other:xPath="_source/insured_ages[0]"
other:sourceField="insured_ages"        other:analyzed="false" />
    <attr name="vehicles"                     xs:type="string"
other:xPath="_source/vehicles"
other:valueFormat="aggregate" other:sourceField="vehicles"
other:analyzed="true"  />
    <attr name="vehicles.0.year"              xs:type="integer"
other:xPath="_source/vehicles[0]/year"
other:sourceField="vehicles.year"       other:analyzed="true"  />
    <attr name="vehicles.0.make"              xs:type="string"
other:xPath="_source/vehicles[0]/make"
other:sourceField="vehicles.make"       other:analyzed="true"  />
    <attr name="vehicles.0.model"             xs:type="string"
other:xPath="_source/vehicles[0]/model"
other:sourceField="vehicles.model"      other:analyzed="true"  />
    <attr name="vehicles.0.body_style"        xs:type="string"
other:xPath="_source/vehicles[0]/body_style"
other:sourceField="vehicles.body_style" other:analyzed="true"  />
```

```
    <input name="rows@next" desc="Internal attribute used for
paging through data."  />
  </rsb:info>

  <rsb:set attr="es_index" value="auto"/>
  <rsb:set attr="es_type"  value="insured"/>

  <rsb:script method="GET">
    <rsb:call op="elasticsearchadoSelect">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

  <rsb:script method="POST">
    <rsb:call op="elasticsearchadoModify">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

  <rsb:script method="MERGE">
    <rsb:call op="elasticsearchadoModify">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

  <rsb:script method="DELETE">
    <rsb:call op="elasticsearchadoModify">
      <rsb:push/>
    </rsb:call>
  </rsb:script>

</rsb:script>
```

SQL Compliance

The Elasticsearch Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Elasticsearch API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

UPSERT Statements

An UPSERT updates a record if it exists and inserts the record if it does not. See UPSERT Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Elasticsearch adapter:

```
SELECT {
  [ TOP <numeric_literal> | DISTINCT ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
    [
      { OFFSET | , }
      <expression>
    ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
```

```
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
     <expression> { = | > | < | >= | <= | <> | != | LIKE | NOT LIKE
| IS NULL | IS NOT NULL | IN | NOT IN | AND | OR | BETWEEN |
CONTAINS | NOT CONTAINS } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Account
```

Rename a column:

```
SELECT "Name" AS MY_Name FROM Account
```

Search data:

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

The Elasticsearch APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IS NULL, IS NOT NULL, IN, NOT IN, AND, OR, BETWEEN, CONTAINS, NOT CONTAINS.

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM Account
```

Return the number of unique items matching the query criteria:

```
SELECT COUNT(DISTINCT Name) FROM Account
```

Return the unique items matching the query criteria:

```
SELECT DISTINCT Name FROM Account
```

Summarize data:

```
SELECT Name, MAX(AnnualRevenue) FROM Account  GROUP BY Name
```

See Aggregate Functions for details.

Sort a result set in ascending order:

```
SELECT Id, Name FROM Account  ORDER BY Name ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM Account WHERE Industry = 'Floppy Disks'
```

COUNT_DISTINCT

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT_DISTINCT(Id) AS DistinctValues FROM Account WHERE
Industry = 'Floppy Disks'
```

COUNT(DISTINCT)

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT(DISTINCT Id) AS DistinctValues FROM Account WHERE
Industry = 'Floppy Disks'
```

AVG

Returns the average of the column values.

```
SELECT Name, AVG(AnnualRevenue) FROM Account WHERE Industry =
'Floppy Disks' GROUP BY Name
```

MIN

Returns the minimum column value.

```
SELECT MIN(AnnualRevenue), Name FROM Account WHERE Industry =
'Floppy Disks' GROUP BY Name
```

MAX

Returns the maximum column value.

```
SELECT Name, MAX(AnnualRevenue) FROM Account WHERE Industry =
'Floppy Disks' GROUP BY Name
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(AnnualRevenue) FROM Account WHERE Industry = 'Floppy
Disks'
```

Predicate Functions

COMMON(expression, cutoff_frequency)

Used to explicitly specify the query type to send and thus will send 'expression' in a common terms query.

Example SQL Query:

```
SELECT * FROM employee WHERE COMMON(about) = 'like to build'
```

Elasticsearch Query:

```
{"common":{"about":{"query":"like to build"}}}
```

expression: The expression to search for.

cutoff_frequency: The cutoff frequency value used to allocate terms to the high or low frequency group. Can be an absolute frequency (>=1) or a relative frequency (0.0 .. 1.0).

FILTER(expression)

Used to explicitly specify the filter context and thus will send 'expression' in a filter context, rather than a query context. A filter context does not affect the calculated scores. This is useful when performing queries where you want part of the filter to be used to calculate scores but filter the results returned (without affecting the score) using additional criteria.

Example SQL Query:

```
SELECT * FROM employee WHERE FILTER(TERM(first_name)) = 'john'
```

Elasticsearch Query:

```
{"filter":{"bool":{"must":{"term":{"first_name":"john"}}}}}
```

expression: Either a column or another function.

GEO_BOUNDING_BOX(column, top_left, bottom_right)

Used to specify a query to filter hits based on a point location using a bounding box.

Example SQL Query:

```
SELECT * FROM cities WHERE GEO_BOUNDING_BOX(location,
'[-74.1,40.73]', '[-71.12,40.01]')
```

Elasticsearch Query:

```
{"bool":{"filter":{"geo_bounding_box":{"location":{"top_left":[-74
.1,40.73],"bottom_right":[-71.12,40.01]}}},"must":[{"match_all":{}
}]}}
```

column: A Geo-point column to perform the GEO_BOUNDING_BOX filter on.

top_left: The top-left coordinates of the bounding box. This value can be an array [shown in example], object of lat and lon values, comma-separated list, or a geohash of a latitude and longitude value.

bottom_right: The bottom-right coordinates of the bounding box. This value can be an array [shown in example], object of lat and lon values, comma-separated list, or a geohash of a latitude and longitude value.

GEO_BOUNDING_BOX(column, top, left, bottom, right)

Used to specify a query to filter hits based on a point location using a bounding box.

Example SQL Query:

```
SELECT * FROM cities WHERE GEO_BOUNDING_BOX(location, -74.1, 40.73,
-71.12, 40.01)
```

Elasticsearch Query:

```
{"bool":{"filter":{"geo_bounding_box":{"location":{"top":-74.1,"le
ft":40.73,"bottom":-71.12,"right":40.01}}},"must":[{"match_all":{}
}]}}
```

column: A Geo-point column to perform the GEO_BOUNDING_BOX filter on.

top: The top coordinate of the bounding box.

left: The left coordinate of the bounding box.

bottom: The bottom coordinate of the bounding box.

right: The right coordinate of the bounding box.

GEO_DISTANCE(column, point_lat_lon, distance)

Used to specify a query to filter documents that include only the hits that exist within a specific distance from a geo point.

Example SQL Query:

```
SELECT * FROM cities WHERE GEO_DISTANCE(location, '40,-70', '12mi')
```

Elasticsearch Query:

```
{"bool":{"filter":{"geo_distance":{"location":"40,-70","distance":
"12mi"}},"must":[{"match_all":{}}]}}
```

column: A Geo-point column to perform the GEO_DISTANCE filter on.

point_lat_lon: The coordinates of a geo point that will be used to measure the distance from. This value can be an array, object of lat and lon values, comma-separated list [shown in example], or a geohash of a latitude and longitude value.

distance: The distance to search within from the specified geo point. This value takes an numeric value along with a distance unit. Common distance units are: mi (miles), yd (yards), ft (feet), in (inch), km (kilometers), m (meters). Please see Elastic documentation for complete list of distance units.

GEO_DISTANCE_RANGE(column, point_lat_lon, from_distance, to_distance)

Used to specify a query to filter documents that include only the hits that exist within a range from a specific geo point.

Example SQL Query:

```
SELECT * FROM cities WHERE GEO_DISTANCE_RANGE(location,
'drn5x1g8cu2y', '10mi', '20mi')
```

Elasticsearch Query:

```
{"bool":{"filter":{"geo_distance_range":{"location":"drn5x1g8cu2y"
,"from":"10mi","to":"20mi"}},"must":[{"match_all":{}}]}}
```

column: A Geo-point column to perform the GEO_DISTANCE_RANGE filter on.

point_lat_lon: The coordinates of a geo point that will be used to measure the range from. This value can be an array, object of lat and lon values, comma-separated list, or a geohash [shown in example] of a latitude and longitude value.

from_distance: The starting distance to calculate the range from the specified geo point. This value takes an numeric value along with a distance unit. Common distance units are: mi (miles), yd (yards), ft (feet), in (inch), km (kilometers), m (meters). Please see Elastic documentation for complete list of distance units.

to_distance: The end distance to calculate the range from the specified geo point. This value takes an numeric value along with a distance unit. Common distance units are: mi (miles), yd (yards), ft (feet), in (inch), km (kilometers), m (meters). Please see Elastic documentation for complete list of distance units.

GEO_POLYGON(column, points)

Used to specify a query to filter hits that only fall within a polygon of points.

Example SQL Query:

```
SELECT * FROM cities WHERE GEO_POLYGON(location,
'[{"lat":40,"lon":-70},{"lat":30,"lon":-80},{"lat":20,"lon":-90}]'
)
```

Elasticsearch Query:

```
{"bool":{"filter":{"geo_polygon":{"location":{"points":[{"lat":40,
"lon":-70},{"lat":30,"lon":-80},{"lat":20,"lon":-90}]}}},"must":[{
"match_all":{}}]}}
```

column: A Geo-point column to perform the GEO_POLYGON filter on.

points: A JSON array of points that make up a polygon. This value can be an array of arrays, object of lat and lon values [shown in example], comma-separated lists, or geohashes of a latitude and longitude value.

GEO_SHAPE(column, type, points [, relation])

Used to specify an inline shape query to filter documents using the geo_shape type to find documents that have a shape that intersects with the query shape.

Example SQL Query:

```
SELECT * FROM shapes WHERE GEO_SHAPE(my_shape, 'envelope', '[[13.0,
53.0], [14.0, 52.0]]
```

Elasticsearch Query:

```
{"bool":{"filter":{"geo_shape":{"my_shape":{"shape":{"type":"envel
ope","coordinates":[[13.0, 53.0], [14.0,
52.0]]}}}},"must":[{"match_all":{}}]}}
```

column: A Geo-shape column to perform the GEO_SHAPE filter on.

type: The type of shape to search for. Valid values: point, linestring, polygon, multipoint, multilinestring, multipolygon, geometrycollection, envelope, and circle. Please see Elastic documentation for further information regarding these shapes.

points: The coordinates for the shape type specified. These coordinates and their structure will vary depending upon the shape type desired. Please see Elastic search documentation for further details.

relation: The name of the spatial relation operator to use at search time. Valid values: intersects (default), disjoint, within, and contains. Please see Elastic documentation for further information regarding spatial relations.

MATCH(column)

Used to explicitly specify the query type to send and thus will send 'column' in a match query.

Example SQL Query:

```
SELECT * FROM employee WHERE MATCH(last_name) = 'SMITH'
```

Elasticsearch Query:
```
{"match":{"last_name":"SMITH"}}
```

column: A column to perform the match query on.

MATCH_PHRASE(column)

Used to explicitly specify the query type to send and thus will send 'column' in a match phrase query.

Example SQL Query:
```
SELECT * FROM employee WHERE MATCH_PHRASE(about) = 'rides
motorbikes'
```

Elasticsearch Query:
```
{"match_phrase":{"about":"rides motorbikes"}}
```

column: A column to perform the match phrase query on.

MATCH_PHRASE_PREFIX(column)

Used to explicitly specify the query type to send and thus will send 'column' in a match phrase prefix query. The match phrase prefix query is the same as a match query except that it allows for prefix matches on the last term in the text.

Example SQL Query:
```
SELECT * FROM employee WHERE MATCH_PHRASE_PREFIX(about) = 'quick
brown f'
```

Elasticsearch Query:
```
{"match_phrase_prefix":{"about":"quick brown f"}}
```

expression: A column to perform the match phrase prefix query on.

TERM(column)

Used to explicitly specify the query type to send and thus will send 'column' in a term query.

Example SQL Query:
```
SELECT * FROM employee WHERE TERM(last_name) = 'jacobs'
```

Elasticsearch Query:
```
{"term":{"last_name":"jacobs"}}
```

column: A column to perform the term query on.

ORDER BY Functions

MAPFIELD(column, data_type)

Used to explicitly specify a mapping (by sending the 'unmapped_type' sort option) for a column that does not have a mapping associated with it, which will enable sorting on the column. By default, if a column does not have a mapping, an exception will be thrown containing an error message similar to: "No mapping found for [column] in order to sort on".

Example SQL Query:

```
SELECT * FROM employee ORDER BY MAPFIELD(start_date, 'long') DESC
```

Elasticsearch Sort:

```
{"start_date":{"order":"desc", "unmapped_type": "long"}}
```

column: The column to perform the order by on.

data_type: The Elasticsearch data type to map the column to.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Name INTO "csv://Account.txt" FROM "Account" WHERE
Industry = 'Floppy Disks'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Name INTO "csv://Account.txt;delimiter=tab" FROM
"Account" WHERE Industry = 'Floppy Disks'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "Floppy Disks");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Account SET Name='Floppy Disks' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

UPSERT Statements

An UPSERT statement will update an existing record or create a new record if an existing record is not identified.

UPSERT Syntax

The UPSERT syntax is the same as for insert. Elasticsearch uses the input provided in the VALUES clause to determine whether the record already exists. If the record does not exist, all columns required to insert the record must be specified. See Data Model for any table-specific information.

```
UPSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "UPSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "Floppy Disks");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:elasticsearch:Server=127.0.0.1;P
ort=9200;",);
String cmd = "DELETE FROM Account WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
```

```
| <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

EXECUTE my_proc second = @p1, first = @p2, third = @p3;

TDV Oracle Eloqua Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to Oracle Eloqua

There are two authentication methods available for connecting to Eloqua: Login and OAuth. The Login method requires you to set Company and to set User, and Password to the credentials you use to log in.

If you do not have access to the username and password or do not wish to require them, you can use OAuth authentication. OAuth is better suited for allowing other users to access their own data. Using login credentials is better suited for accessing your own data.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Bulk Polling Interval	The time interval between requests that check the availability of the bulk query response. The default value is 200 ms.
Bulk Query Timeout	The timeout in minutes for which the provider will wait for a bulk query response. The default value is 25 minutes.

Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Company	The company of the Eloqua account. This field is used to authenticate the user.
Data Retention Duration	The length of time (in hours) that bulk data is stored on the server. Valid values are from 1 hour to 2 weeks. The default value is 24 hours.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Map Data Cards Case Sensitive Match	Whether or not to use case sensitive match in data card mapping.
Map Data Cards Relation Ship	Comma-separated list of the relationships between the Custom Object tables and the Entity tables.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The maximum number of results to return per page from Eloqua.
Password	The password of the Eloqua account used to authenticate.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Eloqua from the provider.
Retry Count	The maximum number of times to retry a Bulk API request that fails due to an HTTP 500 status code (Internal Server Error).
Retry Interval	The time interval between attempts to retry a Bulk API request that failed with an HTTP 500 status code (Internal Server Error).
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
State	An optional value that has meaning for your App.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Bulk API	Whether or not the bulk API is used for retrieving data.
User	The user of the Eloqua account. This field is used to authenticate the user.

Use Simple Names	Boolean determining if simple names should be used for tables and columns.

Bulk Polling Interval

The time interval between requests that check the availability of the bulk query response. The default value is 200 ms.

Data Type

string

Default Value

"200"

Remarks

The time interval between requests that check the availability of the bulk query response. When UseBulkAPI is set, the adapter requests Eloqua to prepare a response to the query. It then waits for the response to be ready by periodically polling the server to check status. This property controls the frequency of polling.

Bulk Query Timeout

The timeout in minutes for which the provider will wait for a bulk query response. The default value is 25 minutes.

Data Type

string

Default Value

"25"

Remarks

The timeout in minutes for which the adapter will wait for a bulk query response. The default value is 25 minutes. When UseBulkAPI is set, the adapter requests Eloqua to prepare a response to the query. It then waits for the response to be ready by periodically polling the server to check status. This property controls the total time the adapter will wait for a response.

Note that this property is very different from Timeout. The Timeout is an inactivity timeout that controls the time to wait for any response. This property controls the total length of time to wait for a bulk query to execute. ;

Callback URL

The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.

Data Type

string

Default Value

""

Remarks

During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

Company

The company of the Eloqua account. This field is used to authenticate the user.

Data Type

string

Default Value

""

Remarks

This field is used to provide authentication for the user to the Eloqua servers.

Data Retention Duration

The length of time (in hours) that bulk data is stored on the server. Valid values are from 1 hour to 2 weeks. The default value is 24 hours.

Data Type

string

Default Value

"24"

Remarks

The length of time (in hours) that bulk data is stored on the server. Valid values are from 1 hour to 2 weeks. The default value is 24 hours.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Oracle Eloqua and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Map Data Cards Case Sensitive Match

Whether or not to use case sensitive match in data card mapping.

Data Type

bool

Default Value

false

Remarks

Whether or not to use case sensitive match in data card mapping. Only has an effect if MapDataCardsRelationShip is set.

Map Data Cards Relation Ship

Comma-separated list of the relationships between the Custom Object tables and the Entity tables.

Data Type

string

Default Value

""

Remarks

Comma-separated list of the relationships between the Custom Object tables and the Entity tables. The format of these relationships is '<custom_object>.<source_field>=<entity_type>.<entity_field>'. For example: Custom_MyCustomObject.Email_Address=Contact.C_EmailAddress

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\Eloqua Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The maximum number of results to return per page from Eloqua.

Data Type

string

Default Value

"500"

Remarks

The Pagesize property affects the maximum number of results to return per page from Eloqua when executing a query. A higher value will return more results per page, but may also cause a timeout exception. The maximum page size supported by Eloqua is 1000.

Password

The password of the Eloqua account used to authenticate.

Data Type

string

Default Value

""

Remarks

This field is used to authenticate against the Eloqua servers.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
```

```
domain\user
```

Readonly

You can use this property to enforce read-only access to Eloqua from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Retry Count

The maximum number of times to retry a Bulk API request that fails due to an HTTP 500 status code (Internal Server Error).

Data Type

string

Default Value

"5"

Remarks

When UseBulkAPI is set to "Auto" or "True", the adapter will attempt to retry any requests that fail due to an HTTP 500 status code (Internal Server Error). This property defintes maximum number of time the adapter will retry a Bulk API request that failed. By default the adapter will retry the request 5 times.

Retry Interval

The time interval between attempts to retry a Bulk API request that failed with an HTTP 500 status code (Internal Server Error).

Data Type

string

Default Value

"200"

Remarks

When UseBulkAPI is set to "Auto" or "True", the adapter will attempt to retry any requests that fail due to an HTTP 500 status code (Internal Server Error). This property defines the time interval between attempts to retry a Bulk API request that failed. The default value is 200 ms.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example

A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

State

An optional value that has meaning for your App.

Data Type

string

Default Value

""

Remarks

An optional value that has meaning for your App..

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Bulk API

Whether or not the bulk API is used for retrieving data.

Data Type

string

Default Value

"Auto"

Remarks

Whether or not the bulk API is used for retrieving data. When UseBulkAPI is set to "True", the adapter will always attempt to use the bulk API. However, there are several restrictions to accessible tables and columns. See Data Model for more information.

When set to "False", the adapter will use the REST API for all requests. When set to "Auto", the adapter will use whichever API is most appropriate for the request. For example, consider the following query:

```
SELECT * FROM Activity_EmailOpen
```

In this case, the adapter will use the Bulk API (because the ContactId is not specified), whereas the following query will use the REST API:

```
SELECT * FROM Activity_EmailOpen ContactId='...'
```

Using the Bulk API starts with the adapter sending a request to Eloqua to prepare a response to the query. It then waits for the response to be ready by periodically polling the server to check status. BulkPollingInterval and BulkQueryTimeout control the frequency and duration of polling respectively.

User

The user of the Eloqua account. This field is used to authenticate the user.

Data Type

string

Default Value

""

Remarks

This field is used to provide authentication for the user to the Eloqua servers.

Use Simple Names

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

Boolean determining if simple names should be used for tables and columns. Eloqua objects can have special characters in names that are normally not allowed in standard databases. UseSimpleNames makes the adapter easier to use with traditional database tools.

Setting UseSimpleNames to true will simplify the names of tables and columns returned making them easier to work with. If set to false, the tables and columns will appear as they do in Eloqua.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Oracle Eloqua Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to Eloqua

OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Configuring the Adapter

The adapter can be configured for OAuth in three ways using the InitiateOAuth property.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and Oracle Eloqua.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

CallbackURL

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from Oracle Eloqua and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If Oracle Eloqua issues a long-lived access token, use the Oracle Eloqua developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

Advanced Settings

Fine-Tuning Data Access

You can use the following properties to gain more granular control over how the adapter surfaces the functionality of the underlying Oracle Eloqua APIs. The adapter uses the bulk API when possible; you can fine-tune the connectivity to the bulk API with the following connection properties:

UseBulkAPI

BulkPollingInterval

BulkQueryTimeout

DataRetentionDuration

Additionally, the following properties are useful to circumvent failed bulk API requests:

RetryCount

RetryInterval

Oracle Eloqua accepts characters for table and column names that must be escaped in SQL. You can set UseSimpleNames to true to report nonalphanumeric characters as underscores. The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

HTTP Proxies

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The Oracle Eloqua Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Oracle Eloqua API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Oracle Eloqua adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
     <expression> { = | != | < | > | >= | <= | AND } [ <expression>
]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Contact
```

Rename a column:

```
SELECT "[First Name]" AS MY_[First Name] FROM Contact
```

Search data:

```
SELECT * FROM Contact WHERE Country = 'U.S.A.';
```

The Oracle Eloqua APIs support the following operators in the WHERE clause: =, !=, <, >, >=, <=, AND.

```
SELECT * FROM Contact WHERE Country = 'U.S.A.';
```

Sort a result set in ascending order:

```
SELECT SalesPerson, [First Name] FROM Contact  ORDER BY [First
Name] ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT SalesPerson, [First Name] INTO "csv://Contact.txt" FROM
"Contact" WHERE Country = 'U.S.A.'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT SalesPerson, [First Name] INTO
"csv://Contact.txt;delimiter=tab" FROM "Contact" WHERE Country =
'U.S.A.'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )
```

```
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Contact ([First Name]) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Contact SET [First Name]='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:eloqua:User=user;Password=passwo
rd;Company=MyCompany",);
String cmd = "DELETE FROM Contact WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
```

```
| <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The Oracle Eloqua Adapter models the Bulk API and the REST API as relational tables, views, and stored procedures. For example, Oracle Eloqua activity types are represented by the corresponding views. Views are tables that cannot be modified.

A full list is available upon customer request.

TDV Facebook Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to Facebook

Facebook uses the OAuth standard to authenticate users. See Connecting to Facebook for an authentication guide.

Posting as a Page

After authenticating to Facebook with your user account, you can post, etc. as one of the pages you manage: Set the AuthenticateAsPage property to the Id of the page you want. You can find the Ids for all pages your account has access to by querying the Pages view.

Automatic Page Authentication

Facebook has made a number of recent changes that require page tokens for most resources owned by a page. This can be troublesome if you manage multiple pages and want to execute the same queries across all pages (such as retrieving Insights). In order to make this work seemlessly with our tools, we have added a way to automatically detect the page token to use. For this to work, simply do not specify the AuthenticateAsPage. Note that the correct page token can only be resolved if the page id is specified as part of the target in the request. This means for some requests you will still need to manually specify AuthenticateAsPage.

Fine-Tuning Data Access

Target: Some Facebook tables can be filtered by a target. For example, to retrieve comments on a video, specify the Id of the video as the target. This property enables you to restrict the results of all queries in the connection to records that match the specified target. You can also specify this restriction per query with the Target column.

AggregateFormat: The adapter returns some columns as a string aggregate. For example, the available likes data for an entity is returned in aggregate. By default the adapter returns aggregate columns in JSON. You can also return aggregates in XML.

Version: Set this property to the Facebook API version if you need to work with a different version than the default.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Aggregate Format	The format aggregate or collection columns should return in.
Authenticate As Page	The name or Id of a page to authenticate as when making requests to Facebook.
Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The maximum number of results to return per page from Facebook.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Facebook from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Target	A default target if none is specified. Used for some tables, such as Comments, where a target may be specified.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Upload Linked Media	Upload linked photos or videos before inserting a new Post.
Version	The Facebook Graph API version to use.

Aggregate Format

The format aggregate or collection columns should return in.

Data Type

string

Default Value

"JSON"

Remarks

The format aggregate or collection columns should return in.

Authenticate As Page

The name or Id of a page to authenticate as when making requests to Facebook.

Data Type

string

Default Value

""

Remarks

The Id of a page to retrieve data from. The page must be managed by the authenticated user; you can obtain the Ids for all such pages by querying the Pages view.

Callback URL

The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.

Data Type

string

Default Value

""

Remarks

During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Facebook and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\Facebook Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The maximum number of results to return per page from Facebook.

Data Type

string

Default Value

""

Remarks

The Pagesize property affects the maximum number of results to return per page from Facebook. Sometimes you may get an error asking you to request less data. The frequency of such errors can be reduced by reducing the pagesize. The maximum pagesize tends to be about 100 per page.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Facebook from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Target

A default target if none is specified. Used for some tables, such as Comments, where a target may be specified.

Data Type

string

Default Value

""

Remarks

A default target if none is specified. Used for some tables, such as Comments, where a target may be specified.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Upload Linked Media

Upload linked photos or videos before inserting a new Post.

Data Type

bool

Default Value

false

Remarks

The UploadLinkedMedia determines whether to upload media before inserting a Post. If set to True, when you attempt to insert a new Post with the Link column, the driver will first attempt to resolve the URL and determine if the URL is referencing a photo or a video. If so, the photo or video will be uploaded first, then a new Post containing the media will be created.If False, then the new Post will be created as a Link Post.

Version

The Facebook Graph API version to use.

Data Type

string

Default Value

"2.12"

Remarks

The Facebook Graph API version to use. Generally this property does not need to be set.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Facebook Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to Facebook

Facebook uses the OAuth authentication standard. OAuth requires the authenticating user to interact with Facebook using the browser. The adapter facilitates this in various ways as described below.

Authenticate to Facebook

After setting InitiateOAuth to GETANDREFRESH, you are ready to connect. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the access token in the connection string. When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

Extracts the access token from the callback URL and authenticates requests.

Refreshes the access token when it expires.

Saves OAuth values in

OAuthSettingsLocation to be persisted across connections.

Requesting Additional Permissions

You may find while using the adapter that Facebook returns an error stating your app does not have permissions to do a certain action. To resolve this, you will need to generate a new OAuth access token with the required permissions. Set the Permissions property in the authentication step. You can find a list of available Facebook permissions here:
http://developers.facebook.com/docs/authentication/permissions/.

Advanced Settings

Create an OAuth App on Facebook

The OAuth user consent flow involves the authenticating user interacting with Facebook using the browser. To facilitate this, the adapter is already registered as an OAuth application, but these credentials are not suitable for Web applications. You may also want to display your own information instead of the CData app's when users log in to grant permissions.

To obtain the OAuth client credentials, follow the steps below:

Log into Facebook and navigate to https://developers.facebook.com/apps.

Create a new app and navigate to your App Settings page. The OAuth client credentials, the App Id and App Secret, are displayed.

Add a website platform on the Settings tab.

If you are making a Desktop application, navigate to Products --> Facebook Login --> Settings, and set the Valid OAuth redirect URIs to http://localhost:33333/.

If you are making a Web application, set the Valid OAuth redirect URIs to the URL you want to be used as a callback URL, where the user will return with the token that verifies that they have granted your app access.

You will need to set the same url under Facebook Login --> Valid OAuth redirect URIs.

Set the App Domain to the domain of the Site URL.

To complete the Connecting to Facebook authentication guide, set the following additional properties:

OAuthClientId: Set this to the App Id in your app settings.

OAuthClientId: Set this to the App Secret in your app settings.

CallbackURL: Set this to the Site URL in your app settings.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

Connecting to Facebook

OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Configuring the Adapter

The adapter can be configured for OAuth in three ways using the InitiateOAuth property.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and Facebook.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

CallbackURL

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from Facebook and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If Facebook issues a long-lived access token, use the Facebook developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

Changes in 2017

In 2017, we have focused on making significant enhancements to how Insights and Ads are handled. We have added numerous views to make the process simpler. To see a complete mapping of which views support which types of insights, please see Insight Mapping.

Page & Post Insight Changes from 2016

In the past, it was required to create insight schemas via CreateInsightSchema. This is no longer necessary with insights where the outputs will always be the same regardless of the type of target. Specifically the new views include:

AdCreatives

AdInsightsActions

GroupMemberships

InsightsByConsumptionType

InsightsByFeedbackType

InsightsByLikeSourceType

InsightsByUnLikeSourceType

InsightsByPaidStatus

InsightsByReactionTotals

InsightsByStoryType

InsightsByTabType

LeadValues

SimpleInsights

Video Insights

We now support views for several types of video insights. The new views listed below can be used for retrieving video insights:

SimpleVideoInsights

VideoInsightsByActionType

VideoInsightsByDistributionType

VideoInsightsByReactionType

Facebook Permission Changes

Facebook has decided to make several permission changes since November 2017. These include 90-day breaking changes, with further 90-day breaking changes made at the end of January 2018. In general, most of these changes have to do with permissions and fall into a couple of categories below.

Page Tokens Required for Most Operations

Facebook now requires a page token to be used for most non-public resources (and some public) off of a page. Most importantly this includes all insights. Previously insights could be retrieved using an appropriate user access token with permissions for the page. Now the AuthenticateAsPage must be specified when retrieving insights.

Note that this does not apply to Ad Insights, only insights created directly off a page or an element owned by a page such as a post by the page.

User Information no Longer Returned from Page Objects

Previously Facebook would allow user data to be returned from public requests. For instance, getting the comments from a post on a page would return information about the user who made the comment. This is no longer available without a Page Access Token. From the Facebook changelog:

User information will not be included in GET responses for any objects owned by (on) a Page unless the request is made with a Page access token.

Automatic Page Token Detection

Due to Facebook's changes requiring a page token for retrieving most objects owned by a page, we have changed the Facebook Adapter to try and resolve the correct page token automatically. Since there is not always identifying information on the owned object itself, this is not guaranteed to work. To use automatic page token detection, simply do not supply an AuthenticateAsPage value. The Facebook Adapter will then build a list of your page tokens and attempt to resolve on each request if a page token should be used, and if so which page token to use. In general this will only work if you are specifying the Target in your request and the ID of the page is is at least part of the specified target.

Group and Event Permission Changes

The permission changes have also affected Facebook Group and Event objects. In general you must now be an administrator of the Group or Event in order to retrieve any data off a Group or Event edge. For instance, SELECT * FROM Videos WHERE Target='GroupId' will no longer work. In addition, GroupMemberships view requires admin access to the group.

Changes to Insight Responses

The Like and Unlike responses have undergone a significant overhaul by Facebook. Pretty much all of the names for the metrics were changed on both Like and Unlike responses. We have had to update our views appropriately since the initial release of 2017. In addition, the Page Paid Status insights were completely redesigned to be in line with the Post insights. Due to this change, we have had to drop the InsightsByPaidStatus view. The new Page Paid Status insights are now merged into SimpleInsights.

Insight Mapping

Below is a mapping of specific Facebook insights, which periods are available for them, what view they can be used from, and what types of targets are available.

View or Stored Procedure	Insight Name	Available Periods	Target Type
SimpleInsights	PAGE_ACTIONS_POST_REACTIONS_ANGER_TOTAL	day	page
SimpleInsights	PAGE_ACTIONS_POST_REACTIONS_HAHA_TOTAL	day	page
SimpleInsights	PAGE_ACTIONS_POST_REACTIONS_LIKE_TOTAL	day	page
SimpleInsights	PAGE_ACTIONS_POST_REACTIONS_LOVE_TOTAL	day	page
SimpleInsights	PAGE_ACTIONS_POST_REACTIONS_SORRY_TOTAL	day	page
SimpleInsights	PAGE_ACTIONS_POST_REACTIONS_WOW_TOTAL	day	page
SimpleInsights	PAGE_CONSUMPTIONS	day, week, days_28	page
SimpleInsights	PAGE_CONSUMPTIONS_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_ENGAGED_USERS	day, week, days_28	page
SimpleInsights	PAGE_FAN_ADDS	day	page
SimpleInsights	PAGE_FAN_ADDS_UNIQUE	day, week, days_28	page

SimpleInsights	PAGE_FAN_REMOVES	day	page
SimpleInsights	PAGE_FAN_REMOVES_UNIQUE	day	page
SimpleInsights	PAGE_FANS	lifetime	page
SimpleInsights	PAGE_FANS_ONLINE	day	page
SimpleInsights	PAGE_FANS_ONLINE_PER_DAY	day	page
SimpleInsights	PAGE_IMPRESSIONS	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_FREQUENCY_DISTRIBUTION	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_ORGANIC	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_ORGANIC_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_PAID	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_PAID_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_VIRAL	day, week, days_28	page

SimpleInsights	PAGE_IMPRESSIONS_VIRAL_FREQUENCY_DISTRIBUTION	day, week, days_28	page
SimpleInsights	PAGE_IMPRESSIONS_VIRAL_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_NEGATIVE_FEEDBACK	day, week, days_28	page
SimpleInsights	PAGE_NEGATIVE_FEEDBACK_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_PLACES_CHECKIN_MOBILE	day, week, days_28	page
SimpleInsights	PAGE_PLACES_CHECKIN_MOBILE_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_PLACES_CHECKIN_TOTAL	day, week, days_28	page
SimpleInsights	PAGE_PLACES_CHECKIN_TOTAL_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS_ORGANIC	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS_ORGANIC_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS_PAID	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS_PAID_UNIQUE	day, week, days_28	page

SimpleInsights	PAGE_POSTS_IMPRESSIONS_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS_VIRAL	day, week, days_28	page
SimpleInsights	PAGE_POSTS_IMPRESSIONS_VIRAL_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_STORIES	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S_AUTOPLAYED	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S_CLICK_TO_PLAY	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S_ORGANIC	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S_PAID	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S_REPEAT_VIEWS	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_COMPLETE_VIEWS_30S_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_REPEAT_VIEWS	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_VIEWS	day, week, days_28	page

SimpleInsights	PAGE_VIDEO_VIEWS_AUTOPLAYED	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_VIEWS_CLICK_TO_PLAY	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_VIEWS_ORGANIC	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_VIEWS_PAID	day, week, days_28	page
SimpleInsights	PAGE_VIDEO_VIEWS_UNIQUE	day, week, days_28	page
SimpleInsights	PAGE_VIEWS	day	page
SimpleInsights	PAGE_VIEWS_LOGIN	day, week	page
SimpleInsights	PAGE_VIEWS_LOGIN_UNIQUE	day, week	page
SimpleInsights	PAGE_VIEWS_LOGOUT	day	page
SimpleInsights	PAGE_VIEWS_UNIQUE	day, week	page
SimpleInsights	POST_CONSUMPTIONS	lifetime	post
SimpleInsights	POST_CONSUMPTIONS_UNIQUE	lifetime	post
SimpleInsights	POST_ENGAGED_USERS	lifetime	post

SimpleInsights	POST_IMPRESSIONS	lifetime	post
SimpleInsights	POST_IMPRESSIONS_FAN	lifetime	post
SimpleInsights	POST_IMPRESSIONS_FAN_PAID	lifetime	post
SimpleInsights	POST_IMPRESSIONS_FAN_PAID_UNIQUE	lifetime	post
SimpleInsights	POST_IMPRESSIONS_FAN_UNIQUE	lifetime	post
SimpleInsights	POST_IMPRESSIONS_ORGANIC	lifetime	post
SimpleInsights	POST_IMPRESSIONS_ORGANIC_UNIQUE	lifetime	post
SimpleInsights	POST_IMPRESSIONS_PAID	lifetime	post
SimpleInsights	POST_IMPRESSIONS_PAID_UNIQUE	lifetime	post
SimpleInsights	POST_IMPRESSIONS_UNIQUE	lifetime	post
SimpleInsights	POST_IMPRESSIONS_VIRAL	lifetime	post
SimpleInsights	POST_IMPRESSIONS_VIRAL_UNIQUE	lifetime	post
SimpleInsights	POST_NEGATIVE_FEEDBACK	lifetime	post

SimpleInsights	POST_NEGATIVE_FEEDBACK_UNIQUE	lifetime	post
SimpleInsights	POST_STORIES	lifetime	post
SimpleInsights	POST_STORYTELLERS	lifetime	post
SimpleInsights	POST_VIDEO_AVG_TIME_WATCHED	lifetime	post
SimpleInsights	POST_VIDEO_COMPLETE_VIEWS_ORGANIC	lifetime	post
SimpleInsights	POST_VIDEO_COMPLETE_VIEWS_ORGANIC_UNIQUE	lifetime	post
SimpleInsights	POST_VIDEO_COMPLETE_VIEWS_PAID	lifetime	post
SimpleInsights	POST_VIDEO_COMPLETE_VIEWS_PAID_UNIQUE	lifetime	post
SimpleInsights	POST_VIDEO_LENGTH	lifetime	post
SimpleInsights	POST_VIDEO_VIEW_TIME	lifetime	post
SimpleInsights	POST_VIDEO_VIEW_TIME_ORGANIC	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_10S	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_10S_AUTOPLAYED	lifetime	post

SimpleInsights	POST_VIDEO_VIEWS_10S_CLICKED_TO_PLAY	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_10S_ORGANIC	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_10S_PAID	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_10S_SOUND_ON	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_10S_UNIQUE	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_ORGANIC	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_ORGANIC_UNIQUE	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_PAID	lifetime	post
SimpleInsights	POST_VIDEO_VIEWS_PAID_UNIQUEPOST_VIDEO_VIEWS_SOUND_ON	lifetime	post
InsightsByConsumptionType	PAGE_CONSUMPTIONS_BY_CONSUMPTION_TYPE	day, week, days_28	page
InsightsByConsumptionType	PAGE_CONSUMPTIONS_BY_CONSUMPTION_TYPE_UNIQUE	day, week, days_28	page
InsightsByConsumptionType	POST_CONSUMPTIONS_BY_CONSUMPTION_TYPE	lifetime	post

InsightsByConsumptionType	POST_CONSUMPTIONS_BY_CONSUMPTION_TYPE_UNIQUE	lifetime	post
InsightsByFeedbackType	PAGE_NEGATIVE_FEEDBACK_BY_TYPE	day, week, days_28	page
InsightsByFeedbackType	PAGE_NEGATIVE_FEEDBACK_BY_TYPE_UNIQUE	day, week, days_28	page
InsightsByFeedbackType	PAGE_POSITIVE_FEEDBACK_BY_TYPE	day, week, days_28	page
InsightsByFeedbackType	PAGE_POSITIVE_FEEDBACK_BY_TYPE_UNIQUE	day, week, days_28	page
InsightsByFeedbackType	POST_NEGATIVE_FEEDBACK_BY_TYPE	lifetime	post
InsightsByFeedbackType	POST_NEGATIVE_FEEDBACK_BY_TYPE_UNIQUE	lifetime	post
InsightsByLikeSourceType	PAGE_FANS_BY_LIKE_SOURCE	day	page
InsightsByLikeSourceType	PAGE_FANS_BY_LIKE_SOURCE_UNIQUE	day	page
InsightsByUnLikeSourceType	PAGE_FANS_BY_UNLIKE_SOURCE	day	page

InsightsByUnLikeSourceType	PAGE_FANS_BY_UNLIKE_SOURCE_UNIQUE	day	page
InsightsByReactionTotals	PAGE_ACTIONS_POST_REACTIONS_TOTAL	day	page
InsightsByReactionTotals	POST_REACTIONS_BY_TYPE_TOTAL	day	page
InsightsByStoryType	PAGE_STORIES_BY_STORY_TYPE	day, week, days_28	page
InsightsByStoryType	PAGE_STORYTELLERS_BY_STORY_TYPE	day, week, days_28	page
InsightsByStoryType	PAGE_IMPRESSIONS_BY_STORY_TYPE	day, week, days_28	page
InsightsByStoryType	PAGE_IMPRESSIONS_BY_STORY_TYPE_UNIQUE	day, week, days_28	page
InsightsByStoryType	POST_IMPRESSIONS_BY_STORY_TYPE	day, week, days_28	page
InsightsByStoryType	POST_IMPRESSIONS_BY_STORY_TYPE_UNIQUE	day, week, days_28	page
InsightsByTabType	PAGE_TAB_VIEWS_LOGIN_TOP_UNIQUE	day, week	page

InsightsByTabType	PAGE_TAB_VIEWS_LOGIN_TOP	day, week	page
InsightsByTabType	PAGE_TAB_VIEWS_LOGOUT_TOP	day	page
SimpleVideoInsights	PAGE_VIDEO_VIEW_TIME	day	page
SimpleVideoInsights	TOTAL_VIDEO_VIEWS	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_VIEWS_UNIQUE	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_VIEWS_AUTOPLAYED	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_VIEWS_CLICKED_TO_PLAY	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_VIEWS_SOUND_ON	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_COMPLETE_VIEWS	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_COMPLETE_VIEWS_UNIQUE	lifetime	video

SimpleVideoInsights	TOTAL_VIDEO_COMPLETE_VIEWS_AUTO_PLAYED	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_COMPLETE_VIEWS_CLICKED_TO_PLAY	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_10S_VIEWS	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_10S_VIEWS_UNIQUE	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_10S_VIEWS_auto_played	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_10S_VIEWS_CLICKED_TO_PLAY	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_10S_VIEWS_SOUND_ON	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_AVG_TIME_WATCHED	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_VIEW_TOTAL_TIME	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_IMPRESSIONS	lifetime	video

SimpleVideoInsights	TOTAL_VIDEO_IMPRESSIONS_UNIQUE	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_IMPRESSIONS_VIRAL_UNIQUE	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_IMPRESSIONS_VIRAL	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_IMPRESSIONS_FAN_UNIQUE	lifetime	video
SimpleVideoInsights	TOTAL_VIDEO_IMPRESSIONS_FAN	lifetime	video
VideoInsightsByActionType	TOTAL_VIDEO_STORIES_BY_ACTION_TYPE	lifetime	video
VideoInsightsByDistributionType	TOTAL_VIDEO_VIEWS_BY_DISTRIBUTION_TYPE	lifetime	video
VideoInsightsByDistributionType	TOTAL_VIDEO_VIEW_TIME_BY_DISTRIBUTION_TYPE	lifetime	video
VideoInsightsByReactionType	TOTAL_VIDEO_REACTIONS_BY_TYPE_TOTAL	lifetime	video
CreateInsightSchema	PAGE_STORYTELLERS_BY_AGE_GENDER	day, week, days_28	page

CreateInsightSchema	PAGE_STORYTELLERS_BY_CITY	day, week, days_28	page
CreateInsightSchema	PAGE_STORYTELLERS_BY_COUNTRY	day, week, days_28	page
CreateInsightSchema	PAGE_STORYTELLERS_BY_LOCALE	day, week, days_28	page
CreateInsightSchema	PAGE_IMPRESSIONS_BY_CITY_UNIQUE	day, week, days_28	page
CreateInsightSchema	PAGE_IMPRESSIONS_BY_COUNTRY_UNIQUE	day, week, days_28	page
CreateInsightSchema	PAGE_IMPRESSIONS_BY_LOCALE_UNIQUE	day, week, days_28	page
CreateInsightSchema	PAGE_IMPRESSIONS_BY_AGE_GENDER_UNIQUE	day, week, days_28	page
CreateInsightSchema	PAGE_PLACES_CHECKINS_BY_AGE_GENDER	day	page
CreateInsightSchema	PAGE_PLACES_CHECKINS_BY_LOCALE	day	page
CreateInsightSchema	PAGE_PLACES_CHECKINS_BY_COUNTRY	day	page

CreateInsightSchema	PAGE_FANS_LOCALE	lifetime	page
CreateInsightSchema	PAGE_FANS_CITY	lifetime	page
CreateInsightSchema	PAGE_FANS_COUNTRY	lifetime	page
CreateInsightSchema	PAGE_FANS_GENDER_AGE	lifetime	page
CreateInsightSchema	PAGE_VIEWS_EXTERNAL_REFERRALS	day	page
CreateInsightSchema	POST_STORIES_BY_ACTION_TYPE	lifetime	post
CreateInsightSchema	POST_STORYTELLERS_BY_ACTION_TYPE	lifetime	post
CreateInsightSchema	PAGE_POSTS_IMPRESSIONS_FREQUENCY_DISTRIBUTION	day, week, days_28	post
CreateInsightSchema	POST_VIDEO_RETENTION_GRAPH	lifetime	post

SQL Compliance

The Facebook Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Facebook API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Facebook adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
```

```
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | > | < | >= | <= | AND | LIKE | NOT
LIKE | IN | NOT IN } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Statuses
```

Rename a column:

```
SELECT "Message" AS MY_Message FROM Statuses
```

Search data:

```
SELECT * FROM Statuses WHERE Target = 'thesimpsons';
```

The Facebook APIs support the following operators in the WHERE clause: =, !=, >, <, >=, <=, AND, LIKE, NOT LIKE, IN, NOT IN.

```
SELECT * FROM Statuses WHERE Target = 'thesimpsons';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT FromName, Message INTO "csv://Statuses.txt" FROM "Statuses"
WHERE Target = 'thesimpsons'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT FromName, Message INTO "csv://Statuses.txt;delimiter=tab"
FROM "Statuses" WHERE Target = 'thesimpsons'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Statuses (Message) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "Sample status message 2");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Statuses SET Message='Sample status message 2'
WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "165189190181_1985180351");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:facebook:InitiateOAuth=GETANDREF
RESH;",);
String cmd = "DELETE FROM Statuses WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "165189190181_1985180351");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

The Facebook Adapter models Facebook APIs as relational Tables, Views, and Stored Procedures. API limitations and requirements are documented in this section; you can use the SupportEnhancedSQL feature, set by default, to circumvent most of these limitations.

A full list is available upon customer request.

TDV Google AdWords Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Google AdWords uses the OAuth authentication standard. The adapter facilitates OAuth in various ways as described below.

Authenticating Google or Google Apps Users

Use this OAuth flow to authenticate individual user accounts. In this OAuth flow, the authenticating user interacts with the browser.

Authenticating to Google AdWords

After setting the following connection properties, you are ready to connect:

InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property.

DeveloperToken: Set this to the developer token of the AdWords account. You can get a developer token by signing up for the AdWords API. To do this, log in to the AdWords site with your MMC account and navigate to Settings -> Account Settings -> AdWords API Center. Apply for API access and wait for Google to contact you via email with more instructions. You should receive a token with a pending status. This token is only for testing and does not allow you to connect to live data. Google should contact you to fill out a questionnaire and once this has been received and approved by Google, you will receive your active developer token.

ClientCustomerId: Set this to the client customer Id of the AdWords account. You can find this value in your AdWords account. This value is not the same as the Id of the MCC account. You need to provide the lowest-level Ids to retrieve data.

When you connect, the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

Extracts the access token from the callback URL and authenticates requests.

Saves OAuth values to be persisted across connections.

Exchanges the refresh token for a new access token when the old access token expires.

Authenticate with a Service Account

You can use this flow to access Google APIs on behalf of users in a domain. A domain administrator must delegate domain-wide access to the service account. In the service account flow, the OAuthAccessToken signifies that the adapter has the same scope of access to Google APIs as the service account.

You will need to register an app to generate a private key. See Advanced Settings to obtain the OAuth values for this flow.

InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property.

OAuthJWTCertType: Set this to "PFXFILE".

OAuthJWTCertPassword: Set this to the password of the .p12 file.

OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

OAuthJWTIssuer: Set this to the email address of the service account.

OAuthJWTCert: Set this to the path to the .p12 file.

OAuthJWTSubject: Set this to the email address of the user for whom the application is requesting delegate access.

DeveloperToken: Set this to the developer token of the AdWords account.

ClientCustomerId: Set this to the client customer Id of the AdWords account.

After setting the following connection properties, click Test Connection. When you connect the adapter completes the OAuth flow for a service account:

Creates and signs the JWT (JSON Web token) with the claim set required by the adapter.

Exchanges the JWT for the access token.

Submits the JWT for a new access token when the token expires.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Client Customer Id	The client customer Id of the AdWords account.
Developer Token	The developer token of the AdWords account.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.

OAuth JWT Cert Password	The password for the OAuth JWT certificate.
OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.
OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Schema	The version of schema to use to get Google AdWords data.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.

Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Client Customer Id

The client customer Id of the AdWords account.

Data Type

string

Default Value

""

Remarks

Together with DeveloperToken, this field is used to authenticate against the Google AdWords servers and is required for use with Google AdWords.

You can find this value in your AdWords account. This value is not the same as the Id of the MCC account. You need to provide the lowest-level Ids to retrieve data.

Developer Token

The developer token of the AdWords account.

Data Type

string

Default Value

""

Remarks

Together with ClientCustomerId, this field is used to authenticate against the Google AdWords servers and is required for use with Google AdWords.

You can get a DeveloperToken by signing up for the AdWords API. To do this, log in to the AdWords site with your MMC account and navigate to Settings -> Account Settings -> AdWords API Center. Apply for API access and wait for Google to contact you via email with more instructions. You should receive a token with a pending status. This token is only for testing and does not allow you to connect to live data. Google should contact you to fill out a questionnaire and once this has been received and approved by Google, you will receive your active developer token.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description

TUNNEL	80	When this is set, the adapter opens a connection to Google AdWords and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State

C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The type of key store containing the JWT Certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.

JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleAdWords Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Schema

The version of schema to use to get Google AdWords data.

Data Type

string

Default Value

"v201802"

Remarks

The version of schema to use to get Google AdWords data.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer

The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.
`Verbosity=4;`

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google AdWords Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Retrieving Google AdWords data

Google AdWords data is organized into various metrics (for example, Conversions, Impressions, and InvalidClicks), which can be grouped by various segments (for example, Date and Device). Additionally, there are attributes that represent fixed data or account settings (for example, CampaignName, AdType, and AdGroupStatus). These are called the Behavior for the field. There are many valid combinations of metrics, segments, and attributes. The adapter surfaces each separate report type in AdWords as a table.

Selecting Data

Unlike most database tables, it is not very helpful to select all metrics, segments, and attributes in a given table. In some cases, it is not even possible to do this since some data conflict with each other when selected. As a result, the adapter thus interprets the SELECT * query to mean a default set of metrics, segments, and attributes are requested. This includes queries that explicitly select all columns to account for tools that use projection when returning results. These default fields represent the default columns exposed through the Google AdWords console. To select nondefault columns, simply explicitly select these columns in the query rather than issuing a SELECT * query. Finally, it is important to note that at least one metric must be included in the request for any data to be returned.

By default, the column names are the values used by the API.

Other Considerations

Certain percentage fields can return the values '< 10%' or '> 90 %'. In order to allow you to use this column as a numeric field, these values are reported as exactly '10' and '90' respectively.

Advanced Settings

Register Your Own Application

The adapter is already registered as an OAuth application and has embedded OAuth credentials, but these credentials are not suitable for the Web application and service account flows. You may also want to display your own information instead of the CData app's when users log in to grant permissions.

To register an app and obtain the OAuth credentials needed for your application, follow the steps below:

User Consent Flow

Log into the Google API Console and open a project. Select the API Manager from the main menu.

In the user consent flow, click Credentials -> Create Credentials -> OAuth Client Id. Click Other.

Service Account Flow

If you are connecting from a service account, follow the steps below:

Log into the Google API Console and open a project. Select the API Manager from the main menu.

Click Create Credentials -> Service Account Key.

In the Service Account menu, select New Service Account or select an existing service account.

If you are creating a new service account, additionally select one or more roles. You can assign primitive roles at the project level in the IAM and Admin section; other roles enable you to further customize access to Google APIs.

In the Key Type section, select the P12 key type.

Create the app to download the key pair. The private key's password is displayed: Set this in

OAuthJWTCertPassword.

In the service accounts section, click Manage Service Accounts and set

OAuthJWTIssuer to the email address displayed in the service account Id field.

You can then follow the connection guide to configure the certificate.

Selecting the API Version

The Google AdWords AdWords API adheres to a deprecation schedule of approximately every 10 months. For backwards compatibility, the adapter maintains APIs that have been deprecated but not sunset; if your application requires a deprecated API, you can set it in the Schema property.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the connection properties needed to authenticate and to connect. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Google AdWords Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Google AdWords API.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google AdWords adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
}

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | IN | NOT IN |
AND } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM CampaignPerformance
```

Rename a column:

```
SELECT "Device" AS MY_Device FROM CampaignPerformance
```

Search data:

```
SELECT * FROM CampaignPerformance WHERE Device = 'Mobile devices
with full browsers';
```

The Google AdWords APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, IN, NOT IN, AND.

```
SELECT * FROM CampaignPerformance WHERE Device = 'Mobile devices
with full browsers';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Clicks, Device INTO "csv://CampaignPerformance.txt" FROM
"CampaignPerformance" WHERE Device = 'Mobile devices with full
browsers'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Clicks, Device INTO
"csv://CampaignPerformance.txt;delimiter=tab" FROM
"CampaignPerformance" WHERE Device = 'Mobile devices with full
browsers'
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

EXECUTE my_proc second = @p1, first = @p2, third = @p3;

Data Model

A full list is available upon customer request.

TDV Google Analytics Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Google Analytics uses the OAuth authentication standard. The adapter facilitates OAuth in various ways as described below; the following OAuth flow requires the authenticating user to interact with Google Analytics, using the browser. For other OAuth flows, see Advanced Settings.

Authenticating to Google Analytics

After setting the following connection properties, you are ready to connect:

Profile: Set this to the Google Analytics profile or view you want to connect to. This value can be retrieved from the Profiles table. If this is not specified, the first Profile returned will be used.

InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property.

When you connect, the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

Extracts the access token from the callback URL and authenticates requests.

Saves OAuth values to be persisted across connections.

Exchanges the refresh token for a new access token when the old access token expires.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Api Version	Specify the API version you want to use.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Include Empty Rows	This connection property can be set only when using the V4 API. If set to false, the provider does not include rows if all the retrieved metrics are equal to zero. The default is true which will include these rows.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.
OAuth JWT Cert Password	The password for the OAuth JWT certificate.
OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.

OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The maximum number of results to return per page from Google Analytics.
Profile	The Google Analytics View (Profile). This can be set to either the Id or website URL for the Profile. If not specified, the first Profile returned will be used.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Sampling Level	The desired sampling level. Can be set to run faster at the cost of accuracy or for higher accuracy but a decrease in query execution speed.

SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Api Version

Specify the API version you want to use.

Data Type

string

Default Value

"V3"

Remarks

Set this property to V3 to use the Google Analytics v3 API or V4 to use the Google Analytics v4 API.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Google Analytics and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Include Empty Rows

This connection property can be set only when using the V4 API. If set to false, the provider does not include rows if all the retrieved metrics are equal to zero. The default is true which will include these rows.

Data Type

string

Default Value

"TRUE"

Remarks

Allowed Values:

TRUE	The provider includes the rows where all the retrieved metrics are equal to zero.
FALSE	The provider does not include the rows where all the retrieved metrics are equal to zero.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State

C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The type of key store containing the JWT Certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.

PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleAnalytics Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The maximum number of results to return per page from Google Analytics.

Data Type

string

Default Value

"10000"

Remarks

The Pagesize property affects the maximum number of results to return per page from Google Analytics when executing a query. A higher value will return more results per page, but may also cause a timeout exception. 10000 is the maximum number of results that may be returned per page from Google.

Profile

The Google Analytics View (Profile). This can be set to either the Id or website URL for the Profile. If not specified, the first Profile returned will be used.

Data Type

string

Default Value

""

Remarks

This value can be retrieved from the Profiles table or will be retrieved automatically if this value is not set.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Sampling Level

The desired sampling level. Can be set to run faster at the cost of accuracy or for higher accuracy but a decrease in query execution speed.

Data Type

string

Default Value

"DEFAULT"

Remarks

Allowed Values:

DEFAULT	Returns response with a sample size that balances speed and accuracy.
FASTER	Available only when using the V3 API. Returns a fast response with a smaller sample size.
HIGHER_PRECISION	Available only when using the V3 API. Returns a more accurate response using a large sample size, but this may result in the response being slower.
SMALL	Similar to FASTER, but for the V4 API.
LARGE	Similar to HIGHER_PRECISION, but for the V4 API.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google Analytics Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to Google

OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Configuring the Adapter

The adapter can be configured for OAuth in three ways using the InitiateOAuth property.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and Google Analytics.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from Google Analytics and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If Google Analytics issues a long-lived access token, use the Google Analytics developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

Retrieving Google Analytics Data

Google Analytics data is organized into various metrics (Sessions, Impressions, AdClicks, etc.), which can be queried over various dimensions (Country, Month, etc.). There are many valid combinations of metrics and dimensions. The adapter surfaces some of the most commonly used combinations as tables for ease of use.

Additionally, the adapter allows you to query all valid combinations, even those not included in the predefined tables, using two schemes: by using the Dimensions and Metrics columns and by defining custom schemas. Refer to Advanced Queries for more information. Below is a guide to getting started with the default tables.

Selecting Dimensions and Measures

The dimension and metrics are clearly defined for each table and can be seen in the Data Model: Simply select the metrics and the dimensions you are interested in. For example, to find the number of sessions in each month, query the Session metric over the Month dimension. This would return 12 rows: one for each month.

```
SELECT Sessions, Month FROM Traffic
```

To separate out the months in each year, include both the month and the year dimensions in the query:

```
SELECT Sessions, Month, Year FROM Traffic
```

SELECT * Query

Unlike most database tables, it is not very helpful to select all metrics and dimensions in a given table. In some cases, it is not even possible to do this since Google Analytics allows a maximum of 7 dimensions and 10 metrics in a single query. The adapter thus interprets the SELECT * query to mean a default set of metrics and dimensions are requested. This includes queries that explicitly select all columns. For schemas with less than 10 metrics, all metrics will be returned. Refer to the individual view's documentation in Data Model to see which fields are the default for each schema.

Advanced Queries

Google Analytics has a very large number of metrics and dimensions that would clutter table definitions, so the table definitions included with the product only list the most commonly used combinations. We offer two alternatives to this design choice: You can use the Dimensions and Metrics columns to request fields that are not in the default table, or you can define your own table.

Using the Dimensions and Metrics Columns

To request additional dimensions or metrics for any existing table, the recommended approach is to define custom schemas; however, you can also set the Dimensions and Metrics inputs in the WHERE clause. Both inputs take a comma-separated list so that you can specify multiple fields at once. The values will be returned in the corresponding Dimensions and Metrics column in the same order that you submitted them. For example, the following query will query the Traffic table for Sessions, the Goal 1 Conversion Rate, and Goal 1 Completions and group these metrics together by the User Age Bracket dimension:

```
SELECT Sessions, Dimensions, Metrics FROM Traffic WHERE
Dimensions='UserAgeBracket' AND
Metrics='Goal1ConversionRate,Goal1Completions'
```

In the results from the query above, the value for UserAgeBracket will be returned in the Dimensions field for each row. The Metrics field will contain a comma-separated value containing the requested metrics for Goal 1.

Defining Custom Schemas

If you routinely need to request fields that are not available on the standard tables that ship with the adapter, you may want to define your own custom schema so that you can more easily query for the data you need. The CreateCustomSchema stored procedure can be used to define entirely new tables, or you can modify existing schemas to add the columns you need.

The stored procedure outputs schema files, which have a simple format that makes them easy to edit directly.

Using the CreateCustomSchema Stored Procedure

The adapter also offers the CreateCustomSchema stored procedure for creating new table definitions. The stored procedure takes a table name, a comma-separated list of metrics, a comma-separated list of dimensions, and an output folder as inputs. Calling it will create a new schema file that you can query like any other table. You will need to set the Location connection property to the folder containing the new script files in order to access them.

The example stored procedure call below persists the columns of the SELECT query in Using the Dimensions and Metrics Columns:

```
EXEC CreateCustomSchema TableName='Traffic',
Dimensions='UserAgeBracket',
Metrics='Sessions,Goal1ConversionRate,Goal1Completions',OutputFold
er='C:\Users\Administrator\Desktop'
```

Edit an Existing Schema Manually

To add fields to an existing schema, open the corresponding .rsd file in the installation directory for the adapter and follow the steps below:

Add an *attr* tag in the <*rsb:info*> section to add a column.

Add the following attributes. Any of the existing fields can serve as an example.

Attribute Name	Attribute Value
name	Set this to a dimension or metric as defined in https://developers.google.com/analytics/devguides/reporting/core/dimsmets .
xs:type	Set this to the data type.
other:dimension	If you want to define a column for a dimension, set this to "true".
other:metric	If you want to define a column for a metric, set this to "true".

To use the new files, set the Location connection property to the folder containing the script files.

Advanced Settings

Fine Tuning Data Access

You can use the following properties to gain greater control over Google Analytics API features and the strategies the adapter uses to surface them:

DefaultFilter: Set this to apply a filter to all queries. Columns you specify in the WHERE clause override values set in this property. See the table-specific information for the tables you are working with to determine columns that can be used in the WHERE clause.

SamplingLevel: Instead of returning every record that matches your query, Google samples results based on the size of a time interval. This property provides several easy options for configuring an appropriate sample level.

Setting Up Advanced OAuth Flows

You can use a service account to connect on behalf of users in a Google Apps domain; in the user consent flow, individual users log into the browser to grant permissions to your application.

To configure the service account flow, you need to register an application to obtain the OAuth JWT values.

You can also register an application to customize the user consent flow by displaying your own information instead of the CData app's.

Register Your Own Application

To register an app and obtain the OAuth credentials, follow the steps below:

Log into the Google API Console and open a project. Select the API Manager from the main menu.

In the user consent flow, click Credentials -> Create Credentials -> OAuth Client Id. Click Other.

After creating the app, the OAuthClientId and OAuthClientSecret are displayed. You can then follow the connection guide by setting these two additional values when you connect.

If you are connecting from a service account, click Service Account Key. In the Service Account menu, select New Service Account or select an existing service account. In the Key Type section, select the P12 key type. Create the app to download the key pair. The private key's password is displayed.

Click Library -> Analytics API -> Enable API.

Authenticate with a Service Account

After setting the following connection properties, you are ready to connect:

InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid manually generating the OAuthAccessToken connection property and repeating the OAuth exchange.

OAuthJWTCertType: Set this to "PFXFILE".

OAuthJWTCertPassword: Set this to the password of the .p12 file.

OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

OAuthJWTIssuer: Set this to the email address of the service account.

OAuthJWTCert: Set this to the path to the .p12 file.

OAuthJWTSubject: Set this to the email address of the user for whom the application is requesting delegate access. Note that delegate access must be granted by an administrator.

Profile: Set this to the Google Analytics profile or view you want to connect to. This value can be retrieved from the Profiles table. If this is not specified, the first Profile returned will be used.

When you connect the adapter completes the OAuth flow for a service account:

Creates and signs the JWT with the claim set required by the adapter.

Exchanges the JWT for the access token.

Submits the JWT for a new access token when the token expires.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the connection properties needed to authenticate and to connect. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Google Analytics Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Google Analytics API.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google Analytics adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
}

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | NOT LIKE
| AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Traffic
```

Rename a column:

```
SELECT "DeviceCategory" AS MY_DeviceCategory FROM Traffic
```

Search data:

```
SELECT * FROM Traffic WHERE Transactions > '0';
```

The Google Analytics APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, AND, OR.

```
SELECT * FROM Traffic WHERE Transactions > '0';
```

Sort a result set in ascending order:

```
SELECT Browser, DeviceCategory FROM Traffic  ORDER BY
DeviceCategory ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Browser, DeviceCategory INTO "csv://Traffic.txt" FROM
"Traffic" WHERE Transactions = '0'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Browser, DeviceCategory INTO
"csv://Traffic.txt;delimiter=tab" FROM "Traffic" WHERE
Transactions = '0'
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

EXECUTE my_proc second = @p1, first = @p2, third = @p3;

Data Model

A full list is available upon customer request.

TDV Google Contacts Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Google Contacts uses the OAuth authentication standard. You can use OAuth to authorize the adapter to access Google APIs on behalf of individual users or on behalf of users in a domain. See Connecting to GoogleContacts for a guide.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.
OAuth JWT Cert Password	The password for the OAuth JWT certificate.
OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.
OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
OAuth Verifier	The verifier code returned from the OAuth authorization URL.
Other	Hidden properties needed only in specific use cases.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.

Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to GoogleContacts from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Firewall Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Google Contacts and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

1.**OFF**: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

2.**GETANDREFRESH**: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

3.**REFRESH**: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.

O	Organization
OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. Note: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. Note: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.

PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. Note: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. Note: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleContacts Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to avoid requiring the user to manually enter OAuth connection properties and allowing the credentials to be shared across connections or processes. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder. Alternatively to specifying a file path, memory storage can be used instead. Memory locations are specified by using a value starting with 'memory://' followed by a unique identifier for that set of credentials (ex: memory://user1). The identifier can be anything you choose but should be unique to the user. Unlike with the file based storage, you must manually store the credentials when closing the connection with memory storage to be able to set them in the connection when the process is started again. The OAuth property values can be retrieved with a query to the sys_connection_props system table. If there are multiple connections using the same credentials, the properties should be read from the last connection to be closed.

OAuth Verifier

The verifier code returned from the OAuth authorization URL.

Data Type

string

Default Value

""

Remarks

The verifier code returned from the OAuth authorization URL. This can be used on systems where a browser cannot be launched such as headless systems.

Authentication on Headless Machines

See the Getting Started section to obtain the OAuthVerifier value.

Set OAuthSettingsLocation along with OAuthVerifier. When you connect, the adapter exchanges the OAuthVerifier for the OAuth authentication tokens and saves them, encrypted, to the specified file. Set InitiateOAuth to GETANDREFRESH automate the exchange.

Once the OAuth settings file has been generated, you can remove OAuthVerifier from the connection properties and connect with OAuthSettingsLocation set.

To automatically refresh the OAuth token values, set OAuthSettingsLocation and additionally set InitiateOAuth to REFRESH.

Other

Hidden properties needed only in specific use cases.

Data Type

string

Default Value

""

Remarks

The properties listed below are available for specific use cases. Normal driver use cases and functionality should not require these properties.

Specify multiple properties in a semicolon-separated list.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

•BASIC: The adapter performs HTTP BASIC authentication.

•DIGEST: The adapter performs HTTP DIGEST authentication.

•NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

•PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

user@domain

domain\user

Readonly

You can use this property to enforce read-only access to GoogleContacts from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----

A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

- Error: Only error messages will be logged.
- Info: Both Error and Info messages will be logged.
- Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

Verbosity=4;

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google Contacts Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at: C:\Program Files\TIBCO\TDS Server 7.0\bin. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to GoogleContacts

You can authorize the adapter to access Google APIs on behalf of individual users or on behalf of a domain. Use the OAuth authentication standard to connect to Google APIs.

Using a User Account to Connect to Google Contacts

This OAuth flow requires the authenticating user to interact with Google Contacts using the browser. The adapter facilitates this in various ways as described below.

Authenticate to Google Contacts

Set InitiateOAuth to "GETANDREFRESH" to avoid manually generating the OAuthAccessToken connection property and repeating the OAuth exchange. When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application.

The adapter then completes the OAuth process:

1.Extracts the access token from the callback URL and authenticates requests.

2.Refreshes the access token when it expires.

3.Saves OAuth values to be persisted across connections. You can configure this file with OAuthSettingsLocation.

Using a Service Account to Connect to Domain-Wide Data

You can use a service account in this OAuth flow to access Google APIs on behalf of users in a domain. A domain administrator must delegate domain-wide access to the service account.

To complete the service account flow, you need to generate a private key in the Google APIs Console. In the service account flow, the adapter obtains an OAuthAccessToken to authenticate that it has the same scope of access to Google APIs as the service account. The adapter exchanges a JSON Web token (JWT) to obtain the access token. The private key is required to sign the JWT.

Generate a Private Key

If you are connecting from a service account, follow the steps below:

1.Log into the Google API Console and open a project. Select the API Manager from the main menu.

2.Click Credentials -> Create Credentials -> Service Account Key.

3.In the Service Account menu, select New Service Account or select an existing service account.

4.If you are creating a new service account, additionally select one or more roles. You can assign primitive roles at the project level in the IAM and Admin section; other roles enable you to further customize access to Google APIs.

5.In the Key Type section, select the P12 key type.

6.Download the key pair. The private key's password is displayed: Set this in OAuthJWTCertPassword.

7.In the Service Account Keys section on the Credentials page, click Manage Service Accounts and set OAuthJWTIssuer to the email address displayed in service account Id.

8.In the API Manager, click Library and enable the Drive, Calendar, and Contacts APIs. To enable an API, click the API and then click Enable API.

Authenticate with a Service Account

After setting the following connection properties, you are ready to connect:

- InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property.
- OAuthJWTCertType: Set this to "PFXFILE".
- OAuthJWTCertPassword: Set this to the password of the .p12 file.

•OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

•OAuthJWTIssuer: Set this to the email address of the service account.

•OAuthJWTCert: Set this to the path to the .p12 file.

•OAuthJWTSubject: Set this to the email address of the user for whom the application is requesting delegate access.

When you connect the adapter completes the OAuth flow for a service account:

1.Creates and signs the JWT with the claim set required by the adapter.

2.Exchanges the JWT for the access token.

3.Submits the JWT for a new access token when the token expires.

Advanced Settings

Register Your Own OAuth Application

The OAuth user consent flow involves the authenticating user to interact with Google using the browser. To facilitate this, the adapter is already registered as an OAuth application, but you may need to configure values specific to your application or organization. You may also want to display your own information instead of the CData app's when users log in to grant permissions.

Follow the steps below to register an app to obtain the credentials needed for your application:

1.Log into the Google API Console and open a project. Select the API Manager from the main menu.

2.Click Credentials -> Create Credentials -> OAuth Client Id. Click Other.

3.Click Library and enable the Drive, Calendar, and Contacts APIs. To enable an API, click the API and then click Enable API.

After creating the app, the OAuthClientId and OAuthClientSecret are displayed. You can then follow the connection guide in Connecting to GoogleContacts by setting these two additional values when you authenticate.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

- **Authentication errors:** Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.
- **Queries time out:** A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Google Contacts Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Google Contacts API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

- Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].
- To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.
- Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

- SELECT
- INTO
- FROM
- JOIN

- WHERE
- GROUP BY
- HAVING
- UNION
- ORDER BY
- LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google Contacts adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | NULLIF ( <expression> , <expression> )
    | COALESCE ( <expression> , ... )
    | CASE <expression>
        WHEN { <expression> | <search_condition> } THEN {
<expression> | NULL } [ ... ]
    [ ELSE { <expression> | NULL } ]
      END
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | AND } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

1.Return all columns:

```
SELECT * FROM My Contacts
```

2. Rename a column:

```
SELECT "Fullname" AS MY_Fullname FROM My Contacts
```

3. Search data:

```
SELECT * FROM My Contacts WHERE Updated = '2017-03-15';
```

4.The Google Contacts APIs support the following operators in the WHERE clause: =, >, <, AND.

```
SELECT * FROM My Contacts WHERE Updated = '2017-03-15';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
boolean ret = stat.execute("SELECT Id, Fullname INTO "csv://c:/My
Contacts.txt" FROM "My Contacts" WHERE Updated = '2017-03-15'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

You can specify other file formats in the URI. The following example exports tab-separated values:

```
Statement stat = conn.createStatement();
boolean ret = stat.execute("SELECT * INTO "My Contacts" IN
'csv://filename=c:/My Contacts.csv;delimiter=tab' FROM "My
Contacts" WHERE Updated = '2017-03-15'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO My Contacts (Fullname) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "Residential Oppty");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE My Contacts SET Fullname='Residential Oppty'
WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "300000002693011");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:googlecontacts:InitiateOAuth=GET
ANDREFRESH;",);
String cmd = "DELETE FROM My Contacts WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "300000002693011");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV Google Calendar Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticate via OAuth Authentication

Use the OAuth authentication standard to connect to Google Calendar. You can authenticate with a user account or with a service account. A service account is required to grant organization-wide access scopes to the adapter. The adapter facilitates these authentication flows as described below.

Authenticate with a User Account

Authenticate with a Service Account

Service accounts have silent authentication, without user authentication in the browser. You can also use a service account to delegate enterprise-wide access scopes to the adapter.

You need to create an OAuth application in this flow. See Creating a Custom OAuth App to create and authorize an app. You can then connect to Google Calendar data that the service account has permission to access.

After setting the following connection properties, you are ready to connect:

•InitiateOAuth: Set this to GETANDREFRESH.

•OAuthClientId: Set this to the Client Id in your app settings.

•OAuthClientSecret: Set this to the Client Secret in your app settings.

•OAuthJWTCertType: Set this to "PEMKEY_FILE".

•OAuthJWTCert: Set this to the path to the .pem file you generated.

•OAuthJWTCertPassword: Set this to the password of the .pem file.

•OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

•OAuthJWTSubject: Set this to your enterprise Id if your subject type is set to "enterprise" or your app user Id if your subject type is set to "user".

When you connect the adapter completes the OAuth flow for a service account.

1.Creates and signs the JWT with the claim set required by the adapter.

2.Exchanges the JWT for the access token.

3.Saves OAuth values in OAuthSettingsLocation to be persisted across connections.

4.Submits the JWT for a new access token when the token expires.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Firewall Password	A password, if authentication is required to connect through a firewall.
Firewall Port	The TCP port for the firewall FirewallServer -- see the description of the FirewallServer option for details.
Firewall Server	Specify a firewall name or IP address to authenticate requested connections, if necessary.
Firewall Type	The type of firewall to connect through.
Firewall User	The user name to authenticate with the firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.
OAuth JWT Cert Password	The password for the OAuth JWT certificate.
OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.
OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
OAuth Verifier	The verifier code returned from the OAuth authorization URL.
Other	Hidden properties needed only in specific use cases.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.

Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Google Calendar from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Firewall Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Google Calendar and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

1.OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

2.GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

3.REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. Note: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. Note: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. Note: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. Note: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.

PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleCalendar Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to avoid requiring the user to manually enter OAuth connection properties and allowing the credentials to be shared across connections or processes. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder. Alternatively to specifying a file path, memory storage can be used instead. Memory locations are specified by using a value starting with 'memory://' followed by a unique identifier for that set of credentials (ex: memory://user1). The identifier can be anything you choose but should be unique to the user. Unlike with the file based storage, you must manually store the credentials when closing the connection with memory storage

to be able to set them in the connection when the process is started again. The OAuth property values can be retrieved with a query to the sys_connection_props system table. If there are multiple connections using the same credentials, the properties should be read from the last connection to be closed.

OAuth Verifier

The verifier code returned from the OAuth authorization URL.

Data Type

string

Default Value

""

Remarks

The verifier code returned from the OAuth authorization URL. This can be used on systems where a browser cannot be launched such as headless systems.

Authentication on Headless Machines

See the Getting Started section to obtain the OAuthVerifier value.

Set OAuthSettingsLocation along with OAuthVerifier. When you connect, the adapter exchanges the OAuthVerifier for the OAuth authentication tokens and saves them, encrypted, to the specified file. Set InitiateOAuth to GETANDREFRESH automate the exchange.

Once the OAuth settings file has been generated, you can remove OAuthVerifier from the connection properties and connect with OAuthSettingsLocation set.

To automatically refresh the OAuth token values, set OAuthSettingsLocation and additionally set InitiateOAuth to REFRESH.

Other

Hidden properties needed only in specific use cases.

Data Type

string

Default Value

""

Remarks

The properties listed below are available for specific use cases. Normal driver use cases and functionality should not require these properties.

Specify multiple properties in a semicolon-separated list.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

•BASIC: The adapter performs HTTP BASIC authentication.

•DIGEST: The adapter performs HTTP DIGEST authentication.

•NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

•PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

user@domain

domain\user

Readonly

You can use this property to enforce read-only access to Google Calendar from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

- Error: Only error messages will be logged.
- Info: Both Error and Info messages will be logged.
- Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

Verbosity=4;

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

- 0 = Error
- 1-2 = Info
- 3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

•1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

•2 - Will log everything included in Verbosity 1 and HTTP headers.

•3 - Will additionally log the body of the HTTP requests.

•4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

•5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google Calendar Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at: C:\Program Files\TIBCO\TDS Server 7.0\bin. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Using OAuth Authentication

Use the OAuth authentication standard to connect to Google Calendar. You can authenticate with a user account or a service account. The adapter facilitates this as described below.

Using a User Account to Authenticate to Google Calendar

The user account flow requires the authenticating user to interact with Google Calendar via the browser.

Embedded Credentials

See Embedded Credentials to connect with the adapter's embedded credentials and skip creating a custom OAuth app.

Custom Credentials

Instead of connecting with the adapter's embedded credentials, you can register an app to obtain the OAuthClientId and OAuthClientSecret.

When to Create a Custom OAuth App

Creating a custom OAuth app is optional as the adapter is already registered with Google Calendar and you can connect with its embedded credentials. You might want to create a custom OAuth app to change the information displayed when users log into the Google Calendar OAuth endpoint to grant permissions to the adapter.

Using a Service Account to Connect to Google Calendar

Service accounts have silent authentication, without user authentication in the browser. You can also use a service account to delegate enterprise-wide access scopes to the adapter.

You need to create an OAuth application in this flow. You can then connect to Google Calendar data that the service account has permission to access. See Custom Credentials for an authentication guide.

Creating a Custom OAuth App

See Creating a Custom OAuth App for a procedure.

Embedded Credentials

Authenticate using the Embedded OAuth Credentials

Desktop Authentication with the Embedded OAuth App

You can connect without setting any connection properties for your user credentials. After setting InitiateOAuth to GETANDREFRESH, you are ready to connect. When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process.

1.Extracts the access token from the callback URL and authenticates requests.

2.Obtains a new access token when the old one expires.

3.Saves OAuth values in OAuthSettingsLocation to be persisted across connections.

Custom Credentials

You can use a custom OAuth app to authenticate a service account or a user account. See Using OAuth Authentication for more information.

Authenticate with a User Account

Desktop Authentication with a Custom OAuth App

Follow the steps below to authenticate with the credentials for a custom OAuth app. See Creating a Custom OAuth App.

Get and Refresh the OAuth Access Token

After setting the following, you are ready to connect:

- OAuthClientId: Set this to the client Id assigned when you registered your app.
- OAuthClientSecret: Set this to the client secret assigned when you registered your app.
- CallbackURL: Set this to http://localhost.

•InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken.

When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

1.Extracts the access token from the callback URL and authenticates requests.

2.Refreshes the access token when it expires.

3.Saves OAuth values in OAuthSettingsLocation to be persisted across connections.

Authenticate with a Service Account

Service accounts have silent authentication, without user authentication in the browser. You can also use a service account to delegate enterprise-wide access scopes to the adapter.

You need to create an OAuth application in this flow. See Creating a Custom OAuth App to create and authorize an app. You can then connect to Google Calendar data that the service account has permission to access.

After setting the following connection properties, you are ready to connect:

•InitiateOAuth: Set this to GETANDREFRESH.

•OAuthClientId: Set this to the Client Id in your app settings.

•OAuthClientSecret: Set this to the Client Secret in your app settings.

•OAuthJWTCertType: Set this to "PEMKEY_FILE".

•OAuthJWTCert: Set this to the path to the .pem file you generated.

•OAuthJWTCertPassword: Set this to the password of the .pem file.

•OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

•OAuthJWTSubject: Set this to the email address of the user for whom the application is requesting delegate access. Note that delegate access must be granted by an administrator.

When you connect the adapter completes the OAuth flow for a service account.

1.Creates and signs the JWT with the claim set required by the adapter.

2.Exchanges the JWT for the access token.

3.Saves OAuth values in OAuthSettingsLocation to be persisted across connections.

4.Submits the JWT for a new access token when the token expires.

Headless Machines

Using OAuth on a Headless Machine

The following sections show how to authenticate a headless server or another machine on which the adapter cannot open a browser. You can authenticate with a user account or with a service account.

Authenticate with a User Account

To authenticate with a user account, you need to authenticate from another machine. Authentication is a two-step process.

1.Instead of installing the adapter on another machine, you can follow the steps below to obtain the OAuthVerifier value. Or, you can install the adapter on another machine and transfer the OAuth authentication values, after you authenticate through the usual browser-based flow.

2.You can then configure the adapter to automatically refresh the access token from the headless machine.

You can follow the headless OAuth authentication flow using the adapter's embedded OAuth credentials or using the OAuth credentials for your custom OAuth app.

Using the Embedded OAuth Credentials

Obtain a Verifier Code

Follow the steps below to authenticate from another machine and obtain the OAuthVerifier connection property:

1.Click the following link to open the Google Calendar OAuth endpoint in your browser.

2.Log in and grant permissions to the adapter. You are then redirected to the callback URL, which contains the verifier code.

3.Save the value of the verifier code. You will set this in the OAuthVerifier connection property.

On the headless machine, set the following connection properties to obtain the OAuth authentication values.

- OAuthVerifier: Set this to the verifier code.
- InitiateOAuth: Set this to REFRESH.
- OAuthSettingsLocation: Set this to persist the encrypted OAuth authentication values to the specified file.

After the OAuth settings file is generated, set the following properties to connect to data:

- OAuthSettingsLocation: Set this to the file containing the encrypted OAuth authentication values. Make sure this file gives read and write permissions to the adapter to enable the automatic refreshing of the access token.
- InitiateOAuth: Set this to REFRESH.

Transfer OAuth Settings

Follow the steps below to install the adapter on another machine, authenticate, and then transfer the resulting OAuth values.

On a second machine, install the adapter and connect with the following properties set:

- OAuthSettingsLocation: Set this to a writable text file.
- InitiateOAuth: Set this to GETANDREFRESH.

Test the connection to authenticate in the browser. The resulting authentication values are written, encrypted, to the path specified by OAuthSettingsLocation. Once you have successfully tested the connection, copy the OAuth settings file to your headless machine.

On the headless machine, set the following connection properties to connect to data:

- InitiateOAuth: Set this to REFRESH.
- OAuthSettingsLocation: Set this to the path to your OAuth settings file. Make sure this file gives read and write permissions to the adapter to enable the automatic refreshing of the access token.

Using the Credentials for a Custom OAuth App

Create a Custom OAuth App

Creating a custom OAuth app is optional in the headless OAuth flow; you can skip creating an app by connecting with the adapter's embedded OAuth credentials. You might want to create a custom OAuth app to change the information displayed when users log into Google Calendar to grant permissions to the adapter.

See Creating a Custom OAuth App for a procedure. You can then follow the procedures below to authenticate and connect to data.

Obtain a Verifier Code

Set the following properties on the headless machine:

- InitiateOAuth: Set this to OFF.
- OAuthClientId: Set this to the Client Id in your app settings.
- OAuthClientSecret: Set this to the Client Secret in your app settings.

You can then follow the steps below to authenticate from another machine and obtain the OAuthVerifier connection property.

1. Call the GetOAuthAuthorizationURL stored procedure with the CallbackURL input parameter set to the exact Redirect URI you specified in your app settings.
2. Open the returned URL in a browser. Log in and grant permissions to the adapter. You are then redirected to the callback URL, which contains the verifier code.
3. Save the value of the verifier code. You will set this in the OAuthVerifier connection property.

On the headless machine, set the following connection properties to obtain the OAuth authentication values:

- OAuthVerifier: Set this to the verifier code.
- OAuthSettingsLocation: Set this to persist the encrypted OAuth authentication values to the specified file.
- InitiateOAuth: Set this to REFRESH.

After the OAuth settings file is generated, set the following properties to connect to data:

•OAuthSettingsLocation: Set this to the file containing the encrypted OAuth authentication values. Make sure this file gives read and write permissions to the provider to enable the automatic refreshing of the access token.

•InitiateOAuth: Set this to REFRESH.

Transfer OAuth Settings

Follow the steps below to install the adapter on another machine, authenticate, and then transfer the resulting OAuth values.

On a second machine, install the adapter and connect with the following properties set:

•OAuthSettingsLocation: Set this to a writable text file.

•InitiateOAuth: Set this to GETANDREFRESH.

•OAuthClientId: Set this to the client Id assigned when you registered your app.

•OAuthClientSecret: Set this to the client secret assigned when you registered your app.

•CallbackURL: Set this to http://localhost.

Test the connection to authenticate. The resulting authentication values are written, encrypted, to the path specified by OAuthSettingsLocation. Once you have successfully tested the connection, copy the OAuth settings file to your headless machine. On the headless machine, set the following connection properties to connect to data:

•InitiateOAuth: Set this to REFRESH.

•OAuthSettingsLocation: Set this to the path to your OAuth settings file. Make sure this file gives read and write permissions to the adapter to enable the automatic refreshing of the access token.

Authenticate with a Service Account

Service accounts have silent authentication, without user authentication in the browser. You can also use a service account to delegate enterprise-wide access scopes to the adapter.

You need to create an OAuth application in this flow. See Creating a Custom OAuth App to create and authorize an app. You can then connect to Google Calendar data that the service account has permission to access.

After setting the following connection properties, you are ready to connect:

•InitiateOAuth: Set this to GETANDREFRESH.

•OAuthClientId: Set this to the Client Id in your app settings.

•OAuthClientSecret: Set this to the Client Secret in your app settings.

•OAuthJWTCertType: Set this to "PEMKEY_FILE".

•OAuthJWTCert: Set this to the path to the .pem file you generated.

•OAuthJWTCertPassword: Set this to the password of the .pem file.

•OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

•OAuthJWTSubject: Set this to your enterprise Id if your subject type is set to "enterprise" or your app user Id if your subject type is set to "user".

When you connect the adapter completes the OAuth flow for a service account.

1.Creates and signs the JWT with the claim set required by the adapter.

2.Exchanges the JWT for the access token.

3.Saves OAuth values in OAuthSettingsLocation to be persisted across connections.

4.Submits the JWT for a new access token when the token expires.

Creating a Custom OAuth App

You can use a custom OAuth app to authenticate a service account or a user account. See Using OAuth Authentication for more information.

Create an OAuth App for User Account Authentication

Follow the procedure below to register an app and obtain the OAuthClientId and OAuthClientSecret.

Create a Custom OAuth App: Desktop

1.Log into the Google API Console.

2.Click Create Project or select an existing project.

3.In the API Manager, click Credentials -> Create Credentials -> OAuth Client Id -> Other.

4.Click Create. The OAuthClientId and OAuthClientSecret are displayed.

5.Click Library -> Google Calendar API -> Enable API.

Create an OAuth App for Service Account Authentication

Follow the steps below to create an OAuth application and generate a private key. You will then authorize the service account.

1.Log into the Google API Console and open a project. Select the API Manager from the main menu.

2.Click Create Credentials -> Service Account Key.

3.In the Service Account menu, select New Service Account or select an existing service account.

4.If you are creating a new service account, additionally select one or more roles. You can assign primitive roles at the project level in the IAM and Admin section; other roles enable you to further customize access to Google APIs.

5.In the Key Type section, select the P12 key type.

6.Create the app to download the key pair. The private key's password is displayed: Set this in OAuthJWTCertPassword.

7.In the service accounts section, click Manage Service Accounts and set OAuthJWTIssuer to the email address displayed in the service account Id field.

8.Click Library -> Google Calendar API -> Enable API.

Advanced Settings

The following sections detail adapter settings that may be needed in advanced integrations.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

- **Authentication errors**: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.
- **Queries time out**: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Google Calendar Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Google Calendar API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

- SELECT
- INTO
- FROM
- JOIN
- WHERE
- GROUP BY
- HAVING
- UNION
- ORDER BY
- LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google Calendar adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | NULLIF ( <expression> , <expression> )
    | COALESCE ( <expression> , ... )
    | CASE <expression>
        WHEN { <expression> | <search_condition> } THEN {
<expression> | NULL } [ ... ]
    [ ELSE { <expression> | NULL } ]
      END
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { =,AND,>,<,=,<=,>= } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

1. Return all columns:

SELECT * FROM MyCalendar

2. Rename a column:

SELECT "Description" AS MY_Description FROM MyCalendar

3. Search data:

SELECT * FROM MyCalendar WHERE Status = 'confirmed';

4. The Google Calendar APIs support the following operators in the WHERE clause: =,AND,>,<,=,<=,>=.

SELECT * FROM MyCalendar WHERE Status = 'confirmed';

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
boolean ret = stat.execute("SELECT Id, Description INTO
"csv://c:/MyCalendar.txt" FROM "MyCalendar" WHERE Status =
'confirmed'");
System.out.println(stat.getUpdateCount()+" rows affected");
You can specify other file formats in the URI. The following
example exports tab-separated values:
Statement stat = conn.createStatement();
boolean ret = stat.execute("SELECT * INTO "MyCalendar" IN
'csv://filename=c:/MyCalendar.csv;delimiter=tab' FROM "MyCalendar"
WHERE Status = 'confirmed'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
```

```
( { <expression> | NULL } [ , ... ] )

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO MyCalendar (Description) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "My Calendar 2");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE MyCalendar SET Description='My Calendar 2'
WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "S");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
```

```
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:googlecalendar:InitiateOAuth=GET
ANDREFRESH;",);
String cmd = "DELETE FROM MyCalendar WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "S");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
```

```
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
Execute a parameterized stored procedure statement:
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV Google Drive Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Google Drive uses the OAuth authentication standard. You can authorize the adapter to connect to Google APIs on behalf of individual users or on behalf of a domain. See Connecting to Google Drive for a guide.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.
OAuth JWT Cert Password	The password for the OAuth JWT certificate.
OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.
OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.
OAuth Verifier	The verifier code returned from the OAuth authorization URL.
Other	Hidden properties needed only in specific use cases.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.

Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Google Drive from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Team Drive Support	Determines whether or not to enable Team Drive support.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Firewall Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Google Drive and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

1.OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

2.GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

3.REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization

OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The type of key store containing the JWT Certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. Note: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. Note: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.

JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. Note: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. Note: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH. Alternatively, this can be held in memory by specifying a value starting with memory://.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleDrive Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to avoid requiring the user to manually enter OAuth connection properties and allowing the credentials to be shared across connections or processes. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder. Alternatively to specifying a file path, memory storage can be used instead. Memory locations are specified by using a value starting with 'memory://' followed by a unique identifier for that set of credentials (ex: memory://user1). The identifier can be anything you choose but should be unique to the user. Unlike with the file based storage, you must manually store the credentials when closing the connection with memory storage to be able to set them in the connection when the process is started again. The OAuth property values can be retrieved with a query to the sys_connection_props system table. If there are multiple connections using the same credentials, the properties should be read from the last connection to be closed.

OAuth Verifier

The verifier code returned from the OAuth authorization URL.

Data Type

string

Default Value

""

Remarks

The verifier code returned from the OAuth authorization URL. This can be used on systems where a browser cannot be launched such as headless systems.

Authentication on Headless Machines

See the Getting Started section to obtain the OAuthVerifier value.

Set OAuthSettingsLocation along with OAuthVerifier. When you connect, the adapter exchanges the OAuthVerifier for the OAuth authentication tokens and saves them, encrypted, to the specified file. Set InitiateOAuth to GETANDREFRESH automate the exchange.

Once the OAuth settings file has been generated, you can remove OAuthVerifier from the connection properties and connect with OAuthSettingsLocation set.

To automatically refresh the OAuth token values, set OAuthSettingsLocation and additionally set InitiateOAuth to REFRESH.

Other

Hidden properties needed only in specific use cases.

Data Type

string

Default Value

""

Remarks

The properties listed below are available for specific use cases. Normal driver use cases and functionality should not require these properties.

Specify multiple properties in a semicolon-separated list.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.

ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

- BASIC: The adapter performs HTTP BASIC authentication.
- DIGEST: The adapter performs HTTP DIGEST authentication.
- NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.
- PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.

NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

user@domain

domain\user

Readonly

You can use this property to enforce read-only access to Google Drive from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Team Drive Support

Determines whether or not to enable Team Drive support.

Data Type

bool

Default Value

false

If you set this property to 'true', you can query from a specific Team Drive using the TeamDriveId as a filter.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

•Error: Only error messages will be logged.

•Info: Both Error and Info messages will be logged.

•Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

Verbosity=4;

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

•0 = Error

•1-2 = Info

•3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

•1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

•2 - Will log everything included in Verbosity 1 and HTTP headers.

•3 - Will additionally log the body of the HTTP requests.

•4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

•5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google Drive Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at: C:\Program Files\TIBCO\TDS Server 7.0\bin. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to Google Drive

The adapter facilitates the following OAuth authentication flows:

- The user consent flow enables individual users to connect to their own data.
- The service account flow enables access to domain-wide data.

Using a User Account to Connect to Google

The adapter facilitates the following OAuth authentication flows:

- The user consent flow enables individual users to connect to their own data.
- The service account flow enables access to domain-wide data.

Authenticate to Google

After setting InitiateOAuth to GETANDREFRESH, you are ready to connect. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property. When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

1. Extracts the access token from the callback URL and authenticates requests.
2. Refreshes the access token when it expires.

3.Saves OAuth values to be persisted across connections. This file can be configured in OAuthSettingsLocation.

Using a Service Account to Connect to Domain-Wide Data

You can use a service account in this OAuth flow to access Google APIs on behalf of users in a domain. A domain administrator can delegate domain-wide access to the service account.

To complete the service account flow, generate a private key in the Google APIs Console. In the service account flow, the adapter exchanges a JSON Web token (JWT) for the OAuthAccessToken. The private key is required to sign the JWT. The OAuthAccessToken authenticates that the adapter has the same permissions granted to the service account

Generate a Private Key

Follow the steps below to generate a private key and obtain the credentials for your application:

1.Log into the Google API Console.

2.Click Create Project or select an existing project.

3.In the API Manager, click Credentials -> Create Credentials -> Service Account Key. In the Service Account menu, select New Service Account or select an existing service account. In the Key Type section, select the P12 key type.

4.Click Create to download the key pair. The private key's password is displayed: Set this in OAuthJWTCertPassword.

5.In the Service Account Keys section on the Credentials page, click Manage Service Accounts and set OAuthJWTIssuer to the email address displayed in service account Id.

6.Click Library -> Google Drive API -> Enable API.

Authenticate with a Service Account

After setting the following connection properties, you are ready to connect:

- InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property.
- OAuthJWTCertType: Set this to "PFXFILE".
- OAuthJWTCertPassword: Set this to the password of the .p12 file.

•OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

•OAuthJWTIssuer: Set this to the email address of the service account.

•OAuthJWTCert: Set this to the path to the .p12 file.

•OAuthJWTSubject: Set this to the email address of the user for whom the application is requesting delegate access.

When you connect the adapter completes the OAuth flow for a service account:

1.Creates and signs the JWT with the claim set required by the adapter.

2.Exchanges the JWT for the access token.

3.Submits the JWT for a new access token when the token expires.

Advanced Settings

The following sections show how to configure adapter features that may be needed in advanced integrations.

Register Your Application

The OAuth user consent flow involves the authenticating user to interact with Google using the browser. To facilitate this, the adapter is already registered as an OAuth application, but you may need to configure values specific to your application or organization. You may also want to display your own information instead of the CData app's when users log in to grant permissions.

Follow the steps below to register an app to obtain the credentials needed for your application:

1.Log into the Google API Console.

2.Click Create Project or select an existing project.

3.In the API Manager, click Credentials -> Create Credentials -> OAuth Client Id.

4.If you are connecting from a desktop application, select Other.

If you are connecting from a Web application, select Web Application. In the Authorized Redirect URIs box, enter the URL you want to be used as a trusted redirect URL, where the user will return with the token that verifies that they have granted your app access.

5.Click Create. The OAuthClientId and OAuthClientSecret are displayed.

6.Click Library -> Google Drive API -> Enable API.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

- Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.
- Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Google Drive Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Google Drive API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

- Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].
- To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.
- Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

- SELECT

- INTO
- FROM
- JOIN
- WHERE
- GROUP BY
- HAVING
- UNION
- ORDER BY
- LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google Drive adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    <column_reference> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | NULLIF ( <expression> , <expression> )
    | COALESCE ( <expression> , ... )
```

```
    | CASE <expression>
        WHEN { <expression> | <search_condition> } THEN {
<expression> | NULL } [ ... ]
    [ ELSE { <expression> | NULL } ]
      END
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | AND | IN | LIKE | >,<,=,<=,>= } [
<expression> ]
    } [ { AND | OR } ... ]
```

Examples

1. Return all columns:

```
SELECT * FROM Files
```

2. Rename a column:

```
SELECT "Name" AS MY_Name FROM Files
```

3. Search data:

```
SELECT * FROM Files WHERE Extension = 'png';
```

4.The Google Drive APIs support the following operators in the WHERE clause: =, AND, IN, LIKE, >,<,=,<=,>=.

```
SELECT * FROM Files WHERE Extension = 'png';
```

5.Sort a result set in ascending order:

```
SELECT Id, Name FROM Files  ORDER BY Name ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
boolean ret = stat.execute("SELECT Id, Name INTO
"csv://c:/Files.txt" FROM "Files" WHERE Extension = 'png'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

You can specify other file formats in the URI. The following example exports tab-separated values:

```
Statement stat = conn.createStatement();
```

```
boolean ret = stat.execute("SELECT * INTO "Files" IN
'csv://filename=c:/Files.csv;delimiter=tab' FROM "Files" WHERE
Extension = 'png'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO Files (Name) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "My File 2");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Files SET Name='My File 2' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "S");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:googledrive:InitiateOAuth=GETAND
REFRESH;",);
String cmd = "DELETE FROM Files WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "S");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV Google BigQuery Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to Google

Google BigQuery uses the OAuth authentication standard. You can authorize the adapter to access Google APIs on behalf on individual users or on behalf of users in a domain.

See Using OAuth for a guide.

Connecting to Google BigQuery

In addition to the OAuth values, specify the DatasetId and ProjectId. You can fine-tune the Google BigQuery functionality surfaced by the adapter with the Advanced Settings.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Allow Large Result Sets	Whether or not to allow large datasets to be stored in temporary tables for large datasets.
Dataset Id	The DatasetId of the dataset you wish to connect to and view tables of.

Destination Table	DestinationTable is entered in the format DestinationProjectId:DestinationDataSet.TableName. Setting DestinationTable allows the query to return very large result sets (more than 128mb of result data) to a permanent destination table.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Google Big Query Options	A comma separated list of Google BigQuery options.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Maximum Billing Tier	The MaximumBillingTier is a positive integer that serves as a multiplier of the basic price per TB. For example, if you set MaximumBillingTier to 2, the maximum cost for that query will be 2x basic price per TB.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.
OAuth JWT Cert Password	The password for the OAuth JWT certificate.

OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.
OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per page from Google BigQuery.
Polling Interval	When DestinationTable is set, or AllowLargeResultSets is true, this determines the polling interval in seconds to check whether the result is ready to be retrieved.
Project Id	The ProjectId of the billing project for executing jobs.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.

Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Query Passthrough	This option passes the query to Google BigQuery as-is.
Readonly	You can use this property to enforce read-only access to Google BigQuery from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Temp Table Dataset	The name of the dataset that will contain temporary tables when executing queries with large result sets.
Temp Table Expiration Time	Time, in seconds until the temporary table expires. Set to 0 to have the table never expire.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Legacy SQL	Specifies whether to use BigQuery's legacy SQL dialect for this query. By default, Standard SQL will be used.

Allow Large Result Sets

Whether or not to allow large datasets to be stored in temporary tables for large datasets.

Data Type

bool

Default Value

true

Remarks

Whether or not to allow large datasets to be stored in temporary tables for large datasets.

Dataset Id

The DatasetId of the dataset you wish to connect to and view tables of.

Data Type

string

Default Value

""

Remarks

The DatasetId of the dataset you wish to connect to and view tables of. You can find this value in Google BigQuery, after selecting a project.

After connecting to an initial dataset, you can query the Datasets view to discover the Ids for all datasets in a project. Along with ProjectId.

Destination Table

DestinationTable is entered in the format DestinationProjectId:DestinationDataSet.TableName. Setting DestinationTable allows the query to return very large result sets (more than 128mb of result data) to a permanent destination table.

Data Type

string

Default Value

""

Remarks

Queries that return large results will take longer to execute, even if the result set is small, and are subject to additional limitations:

A resultset destination table will be created.

You cannot specify a top-level ORDER BY clause.

Large data volumes cannot use the TOP function.

Note: *The default write mode for this table is WRITE_TRUNCATE, so each query will drop the existing table and write the new result set. It is suggested each connection has a different DestinationTable specified.*

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Google BigQuery and traffic flows back and forth through the proxy.

SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Google Big Query Options

A comma separated list of Google BigQuery options.

Data Type

string

Default Value

""

Remarks

A list of Google BigQuery options:

Option	Description
gbqoImplicitJoinAsUnion	This option will prevent the driver from converting an IMPLICIT JOIN into a CROSS JOIN as expected by SQL92. Instead, it will leave it as an IMPLICIT JOIN, which Google BigQuery will execute as a UNION ALL.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Maximum Billing Tier

The MaximumBillingTier is a positive integer that serves as a multiplier of the basic price per TB. For example, if you set MaximumBillingTier to 2, the maximum cost for that query will be 2x basic price per TB.

Data Type

string

Default Value

""

Remarks

Limits the billing tier for this job. Queries that have resource usage beyond this tier will fail (without incurring a charge). If unspecified, this will be set to your project default. If your query is too compute intensive for BigQuery to complete at the standard per TB pricing tier, BigQuery returns a billingTierLimitExceeded error and an estimate of how much the query would cost. To run the query at a higher pricing tier, pass a new value for maximumBillingTier as part of the query request. The maximumBillingTier is a positive integer that serves as a multiplier of the basic price per TB. For example, if you set maximumBillingTier to 2, the maximum cost for that query will be 2x basic price per TB.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State

C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The type of key store containing the JWT Certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.

PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleBigQuery Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per page from Google BigQuery.

Data Type

string

Default Value

"100000"

Remarks

The pagesize can control the number of results returned per page from Google BigQuery. Setting a higher pagesize will cause more data to come back in a single HTTP request, but may take longer to execute. Setting a smaller pagesize will increase the number of HTTP requests to get all the data, but is generally recommended to ensure timeout exceptions do not occur.

Polling Interval

When DestinationTable is set, or AllowLargeResultSets is true, this determines the polling interval in seconds to check whether the result is ready to be retrieved.

Data Type

string

Default Value

"2"

Remarks

Only applicable when DestinationTable is set or AllowLargeResultSets is true. This property determines how long to wait between checking whether or not the query's results are ready. Very large resultsets or complex queries may take longer to process, and a low polling interval may result in may unnecessary requests being made to check the query status.

Project Id

The ProjectId of the billing project for executing jobs.

Data Type

string

Default Value

""

Remarks

The ProjectId of the billing project for executing jobs. You can obtain the project Id in the Google APIs console: In the main menu, click API Project and copy the Id. For example: *psychic-valve-137816*. (Note that your domain name is not part of the Id.)

After connecting to a project, you can determine the other projects accessible to you by querying the Projects view. Set this property to the value of the NumericId field returned by this view.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Query Passthrough

This option passes the query to Google BigQuery as-is.

Data Type

bool

Default Value

false

Remarks

This option passes the query to Google BigQuery as-is.

Readonly

You can use this property to enforce read-only access to Google BigQuery from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----

The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Temp Table Dataset

The name of the dataset that will contain temporary tables when executing queries with large result sets.

Data Type

string

Default Value

"_CDataTempTableDataset"

Remarks

The name of the dataset that will contain temporary tables when executing queries with large result sets.

Temp Table Expiration Time

Time, in seconds until the temporary table expires. Set to 0 to have the table never expire.

Data Type

string

Default Value

"36000"

Remarks

Time, in seconds until the temporary table expires. Set to 0 to have the table never expire. The minimum value is 3600 (One Hour).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Legacy SQL

Specifies whether to use BigQuery's legacy SQL dialect for this query. By default, Standard SQL will be used.

Data Type

bool

Default Value

false

Remarks

If set to true, the query will use BigQuery's Legacy SQL dialect to rebuild the query. If set to false, the query will use BigQuery's standard SQL: https://cloud.google.com/bigquery/sql-reference/.
When UseLegacySQL is set to false, the values of AllowLargeResultSets is ignored. The query will be run as if AllowLargeResultSets is true.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google BigQuery Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Using OAuth

Google supports the following OAuth flows:

The user consent flow enables individual users to connect to their own data.

The service account flow enables access to domain-wide data.

The adapter simplifies both of these flows. See the following sections to set the necessary connection properties and complete authentication.

In addition to the OAuth values, you will need to provide the DatasetId and ProjectId.

Using a User Account to Connect to Google BigQuery

This OAuth flow requires the authenticating user to interact with Google using the browser. The adapter facilitates this in various ways as described below.

Authenticate to Google BigQuery from a Desktop Application

After setting the following connection properties, you are ready to connect:

InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid manually generating the OAuthAccessToken connection property and repeating the OAuth exchange.

DatasetId: Set this to the Id of the dataset you want to connect to.

ProjectId: Set this to the Id of the project you want to connect to.

When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

Extracts the access token from the callback URL and authenticates requests.

Refreshes the access token when it expires.

Saves OAuth values to be persisted across connections. This file can be configured in

OAuthSettingsLocation.

Authenticate from a Web Application

To obtain an OAuthAccessToken, you need to register an app and set the following connection properties. For a guide to registering an app, see Advanced Settings.

OAuthClientId: Set this to the client Id in your app settings.

OAuthClientSecret: Set this to the client secret in your app settings.

When connecting via a Web application, or if the adapter is not authorized to open a browser window, you exchange a verifier code for the access token:

Call GetOAuthAuthorizationURL

. The stored procedure returns the URL of the OAuth endpoint.

Log in at the OAuth endpoint and authorize the application. You are redirected back via the callback URL.

The verifier code is appended to the callback URL in a query string parameter named "code". Extract the verifier code.

Call GetOAuthAccessToken

.

To connect to data, set the following connection properties:

OAuthAccessToken

DatasetId

ProjectId

When the access token expires, call RefreshOAuthAccessToken.

Using a Service Account to Connect to Domain-Wide Data

You can use a service account in this OAuth flow to access Google APIs on behalf of users in a domain. A domain administrator can delegate domain-wide access to the service account.

To complete the service account flow, generate a private key in the Google APIs Console. In the service account flow, the adapter exchanges a JSON Web token (JWT) for the OAuthAccessToken. The OAuthAccessToken authenticates that the adapter has the same permissions granted to the service account. The private key is required to sign the JWT.

Generate a Private Key

Follow the steps below to obtain the credentials for your application:

Log into the Google API Console.

Click Create Project or select an existing project.

In the API Manager, click Credentials -> Create Credentials -> Service Account Key. In the Service Account menu, select New Service Account or select an existing service account. In the Key Type section, select the P12 key type.

Click Create to download the key pair. The private key's password is displayed: Set this in

OAuthJWTCertPassword.

In the Service Account Keys section on the Credentials page, click Manage Service Accounts and set

OAuthJWTIssuer to the email address displayed in the Service Account Id field.

Click Library -> BigQuery API -> Enable API.

Authenticate with a Service Account

After setting the following connection properties, you are ready to connect:

InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid manually generating the OAuthAccessToken connection property and repeating the OAuth exchange.

OAuthJWTCertType: Set this to "PFXFILE".

OAuthJWTCertPassword: Set this to the password of the .p12 file.

OAuthJWTCertSubject: Set this to "*" to pick the first certificate in the certificate store.

OAuthJWTIssuer: Set this to the email address of the service account.

OAuthJWTCert: Set this to the path to the .p12 file.

OAuthJWTSubject: Set this to the email address of the user for whom the application is requesting delegate access.

DatasetId: Set this to the Id of the dataset you want to connect to.

ProjectId: Set this to the Id of the project you want to connect to.

When you connect the adapter completes the OAuth flow for a service account:

Creates and signs the JWT with the claim set required by the adapter.

Exchanges the JWT for the access token.

Submits the JWT for a new access token when the token expires.

Advanced Settings

The following sections detail adapter settings that may be needed in advanced integrations.

Configuring the SQL Dialect

The adapter enables standards-based access to Google BigQuery through SQL-92 or the Google BigQuery syntax:

GoogleBigQueryOptions: Generally, Google BigQuery SQL is a superset of SQL-92; that is, it is SQL-92 compliant but offers several additional functions. However, some differences in the dialects can cause unexpected behavior. For example, Google BigQuery interprets implicit joins as a UNION ALL statement, while implicit joins in SQL-92 are interpreted as CROSS JOIN.

UseLegacySQL: By default, the adapter translates SQL-92 statements to Google BigQuery's Legacy SQL dialect. Set this to false to translate to Google BigQuery's standard SQL. This can be useful to access functionality specific to one Google BigQuery syntax or to avoid an error when an SQL-92 query cannot be translated to the underlying Google BigQuery syntax.

QueryPassthrough: Set this true to bypass the SQL engine of the adapter and execute queries directly to Google BigQuery.

Saving Result Sets

Large result sets must be saved in a temporary or permanent table. You can use the following properties to control table persistence:

Accessing Temporary Tables

You can use the following properties to manage temporary tables:

AllowLargeResultSets: Store large result sets in temporary tables in a special dataset. If this is not set, an error is returned for result sets bigger than a certain size.

TempTableDataset: Change the name of the dataset where temporary tables are saved.

TempTableExpirationTime: Drop a temporary table the specified time after its creation.

Accessing Persistent Tables

Persist result sets larger than 128MB in a permanent table. To do so, you can set DestinationTable to the qualified name of the table. For example, "DestinationProjectId:DestinationDataSet.TableName". Subsequent queries replace the table with the new result set.

Limiting Billing

Set MaximumBillingTier to override your project limits on the maximum cost for any given query in a connection.

Registering Your Application

The OAuth user consent flow involves the authenticating user to interact with Google using the browser. To facilitate this, the adapter is already registered as an OAuth application, but you may need to configure values specific to your application or organization. You may also want to display your own information instead of the CData app's when users log in to grant permissions.

Log into the Google API Console.

Click Create Project or select an existing project.

Open the API Manager from the main menu and click Credentials -> Create Credentials -> OAuth Client Id.

If you are connecting from a desktop application, select Other.

If you are connecting from a Web application, select Web Application. In the Authorized Redirect URIs box, enter the URL you want to be used as a trusted redirect URL, where the user will return with the token that verifies that they have granted your app access.

Click Create. The OAuthClientId and OAuthClientSecret are displayed.

Click Library -> BigQuery API -> Enable API.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Google BigQuery Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Google BigQuery API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

Google BigQuery API Syntax

The Google BigQuery API offers additional SQL operators and functions. A complete list of the available syntax is located at:
https://cloud.google.com/bigquery/query-reference

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google BigQuery adapter:

```
SELECT {
  [ TOP <numeric_literal> | DISTINCT ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  } [ , ... ]
  [ [
      INNER | { { LEFT | RIGHT | FULL  } [ OUTER ] }
    ] JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [ HAVING <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
```

```
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | NOT LIKE
| IN | NOT IN | IS NULL | IS NOT NULL | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM publicdata:samples.github_nested
```

Rename a column:

```
SELECT "repository.name" AS MY_repository.name FROM
publicdata:samples.github_nested
```

Search data:

```
SELECT * FROM publicdata:samples.github_nested WHERE
repository.name = 'EntityFramework';
```

The Google BigQuery APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR.

```
SELECT * FROM publicdata:samples.github_nested WHERE
repository.name = 'EntityFramework';
```

Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM publicdata:samples.github_nested
```

Return the unique items matching the query criteria:

```
SELECT DISTINCT repository.name FROM
publicdata:samples.github_nested
```

Summarize data:

```
SELECT repository.name, MAX(repository.watchers) FROM
publicdata:samples.github_nested  GROUP BY repository.name
```

See Aggregate Functions for details.

Retrieve data from multiple tables.

```
SELECT * FROM CRMAccounts INNER JOIN ERPCustomers ON
CRMAccounts.BillingState = ERPCustomers.BillingState
```

See JOIN Queries for details.

Sort a result set in ascending order:

```
SELECT actor.attributes.email, repository.name FROM
publicdata:samples.github_nested  ORDER BY repository.name ASC
```

Aggregate Functions

Google BigQuery API Syntax

The Google BigQuery API offers additional SQL operators and functions. A complete list of the available syntax is located at:
https://cloud.google.com/bigquery/query-reference

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM publicdata:samples.github_nested WHERE
repository.name = 'EntityFramework'
```

COUNT(DISTINCT)

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT(DISTINCT actor.attributes.email) AS DistinctValues
FROM publicdata:samples.github_nested WHERE repository.name =
'EntityFramework'
```

AVG

Returns the average of the column values.

```
SELECT repository.name, AVG(repository.watchers) FROM
publicdata:samples.github_nested WHERE repository.name =
'EntityFramework' GROUP BY repository.name
```

MIN

Returns the minimum column value.

```
SELECT MIN(repository.watchers), repository.name FROM
publicdata:samples.github_nested WHERE repository.name =
'EntityFramework' GROUP BY repository.name
```

MAX

Returns the maximum column value.

```
SELECT repository.name, MAX(repository.watchers) FROM
publicdata:samples.github_nested WHERE repository.name =
'EntityFramework' GROUP BY repository.name
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(repository.watchers) FROM
publicdata:samples.github_nested WHERE repository.name =
'EntityFramework'
```

CORR

Returns the Pearson correlation coefficient of a set of number pairs.

```
SELECT repository.name, CORR(repository.watchers, repository.size)
FROM publicdata:samples.github_nested
```

COVAR_POP

Computes the population covariance of the values computed by a set of number pairs.

```
SELECT repository.name, COVAR_POP(repository.watchers,
repository.size) FROM publicdata:samples.github_nested
```

COVAR_SAMP

Computes the sample covariance of the values computed by a set of number pairs.

```
SELECT repository.name, COVAR_SAMP(repository.watchers,
repository.size) FROM publicdata:samples.github_nested
```

NTH

Returns the nth sequential value in the scope of the function, where n is a constant. The NTH function starts counting at 1, so there is no zeroth term. If the scope of the function has less than n values, the function returns NULL.

```
SELECT repository.name, NTH(n, actor.attributes.email) FROM
publicdata:samples.github_nested
```

STDDEV

Returns the standard deviation of the computed values. Rows with a NULL value are not included in the calculation.

```
SELECT repository.name, STDDEV(repository.watchers) FROM
publicdata:samples.github_nested
```

JOIN Queries

The adapter supports the complete join syntax in Google BigQuery. Google BigQuery supports inner joins, outer joins, and cross joins. The default is inner. Multiple join operations are supported.

```
SELECT field_1 [..., field_n] FROM
   table_1 [[AS] alias_1]
 [[INNER|[FULL|RIGHT|LEFT] OUTER|CROSS] JOIN [EACH]
   table_2 [[AS] alias_2]
  [ON join_condition_1 [... AND join_condition_n]]
 ]+
```

Note that the default join is an inner join. The following limitations exist on joins in Google BigQuery:

Cross joins must not contain an ON clause.

Normal joins require that the right-side table must contain less than 8 MB of compressed data. If you are working with tables larger than 8 MB, use the EACH modifier. Note that EACH cannot be used in cross joins.

Projection Functions

ANY_VALUE(expression)

Returns any value from the input or NULL if there are zero input rows. The value returned is non-deterministic, which means you might receive a different result each time you use this function.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to retrieve a value from.

APPROX_COUNT_DISTINCT(expression)

Returns the approximate result for COUNT(DISTINCT expression). The value returned is a statistical estimate-not necessarily the actual value. This function is less accurate than COUNT(DISTINCT expression), but performs better on huge input.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the approximate count distinct on.

APPROX_QUANTILES(expression, number)

Returns the approximate boundaries for a group of expression values, where number represents the number of quantiles to create. This function returns an array of number + 1 elements, where the first element is the approximate minimum and the last element is the approximate maximum.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the approximate quantiles on.

number: The number of quantiles to create.

APPROX_TOP_COUNT(expression, number)

Returns the approximate top elements of expression. The number parameter specifies the number of elements returned.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the approximate top count on.

number: The number of elements to be returned.

APPROX_TOP_SUM(expression, weight, number)

Returns the approximate top elements of expression, based on the sum of an assigned weight. The number parameter specifies the number of elements returned. If the weight input is negative or NaN, this function returns an error.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the approximate top sum on.

weight: The assigned weight.

number: The number of elements to be returned.

ARRAY_AGG(expression)

Returns an ARRAY of expression values.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression values to generate an array from.

ARRAY_CONCAT_AGG(expression1[, expression2][,...])

Concatenates elements from expression of type ARRAY, returning a single ARRAY as a result. This function ignores NULL input arrays, but respects the NULL elements in non-NULL input arrays (an error is raised, however, if an array in the final query result contains a NULL element).
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression1: The first expression to concatenate.

expression2: The first expression to concatenate.

AVG(DISTINCT expression)

Returns the average on non-null values. Each distinct value of expression is aggregated only once into the result.

expression: The expression to use to compute the average.

BIT_AND(numeric_expression)

Returns the result of a bitwise AND operation between each instance of numeric_expr across all rows. NULL values are ignored. This function returns NULL if all instances of numeric_expr evaluate to NULL.

numeric_expression: The numeric expression to perform the bitwise operation.

BIT_COUNT(expression)

The input, expression, must be an integer or BYTES. Returns the number of bits that are set in the input expression. For integers, this is the number of bits in two's complement form.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the bit count operation on.

BIT_OR(numeric_expression)

Returns the result of a bitwise OR operation between each instance of numeric_expr across all rows. NULL values are ignored. This function returns NULL if all instances of numeric_expr evaluate to NULL.

numeric_expression: The numeric expression to perform the bitwise operation.

BIT_XOR(numeric_expression)

Returns the result of a bitwise XOR operation between each instance of numeric_expr across all rows. NULL values are ignored. This function returns NULL if all instances of numeric_expr evaluate to NULL.

numeric_expression: The numeric expression to perform the bitwise operation.

CORR(numeric_expression, numeric_expression)

Returns the Pearson correlation coefficient of a set of number pairs.

numeric_expression: The first series.

numeric_expression: The second series.

COUNTIF(expression)

Returns the count of TRUE values for expression. Returns 0 if there are zero input rows or expression evaluates to FALSE for all rows.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to evaluate.

COVAR_POP(numeric_expression1, numeric_expression2)

Computes the population covariance of the values computed by numeric_expression1 and numeric_expression2.

numeric_expression: The first series.

numeric_expression: The second series.

COVAR_SAMP(numeric_expression1, numeric_expression2)

Computes the sample covariance of the values computed by numeric_expression1 and numeric_expression2.

numeric_expression: The first series.

numeric_expression: The second series.

CURRENT_DATE()

Returns a human-readable string of the current date in the format %Y-%m-%d.

CURRENT_TIME()

Returns a human-readable string of the server's current time in the format %H:%M:%S.

CURRENT_TIMESTAMP()

Returns a TIMESTAMP data type of the server's current time in the format %Y-%m-%d %H:%M:%S.

FIRST(column)

Returns the first sequential value in the scope of the function.
Note: this function is only available when UseLegacySQL=True.

column: Any column expression.

FIRST_VALUE(value_expression)

Returns the value of the value_expression for the first row in the current window frame.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

value_expression: Any value expression

GROUP_CONCAT(string_expression [, separator])

Concatenates multiple strings into a single string, where each value is separated by the optional separator parameter. If separator is omitted, returns a comma-separated string.
Note: this function is only available when UseLegacySQL=True.

string_expression: The string expression to concat.

separator: The separator.

GROUP_CONCAT_UNQUOTED(string_expression [, separator])

Concatenates multiple strings into a single string, where each value is separated by the optional separator parameter. If separator is omitted, BigQuery returns a comma-separated string. Unlike GROUP_CONCAT, this function will not add double quotes to returned values that include a double quote character. For example, the string a"b would return as a"b.
Note: this function is only available when UseLegacySQL=True.

string_expression: The string expression to concat.

separator: The separator.

LAST(column)

Returns the last sequential value in the scope of the function.
Note: this function is only available when UseLegacySQL=True.

column: Any column expression

LAST_VALUE(value_expression)

Returns the value of the value_expression for the last row in the current window frame.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

value_expression: Any value expression

LOGICAL_AND(expression)

Returns the logical AND of all non-NULL expressions. Returns NULL if there are zero input rows or expression evaluates to NULL for all rows.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the logical AND on.

LOGICAL_OR(expression)

Returns the logical OR of all non-NULL expressions. Returns NULL if there are zero input rows or expression evaluates to NULL for all rows.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to perform the logical OR on.

NEST(expression)

Aggregates all values in the current aggregation scope into a repeated field. For example, the query SELECT x, NEST(y) FROM ... GROUP BY x returns one output record for each distinct x value, and contains a repeated field for all y values paired with x in the query input. The NEST function requires a GROUP BY clause.
Note: this function is only available when UseLegacySQL=True.

expression: Any expression.

NOW()

Returns the current UNIX timestamp in microseconds.
Note: this function is only available when UseLegacySQL=True.

NTH(n, field)

Returns the nth sequential value in the scope of the function, where n is a constant. The NTH function starts counting at 1, so there is no zeroth term. If the scope of the function has less than n values, the function returns NULL.
Note: this function is only available when UseLegacySQL=True.

n: The nth sequential value.

field: The column name.

NTH_VALUE(value_expression, constant_integer_expression)

Returns the value of value_expression at the Nth row of the current window frame, where Nth is defined by constant_integer_expression. Returns NULL if there is no such row.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

value_expression: Any value expression.

constant_integer_expression: The nth sequential value.

QUANTILES(expression [, buckets])

Computes approximate minimum, maximum, and quantiles for the input expression. NULL input values are ignored. Empty or exclusively-NULL input results in NULL output. The number of quantiles computed is controlled with the optional buckets parameter, which includes the minimum and maximum in the count.
Note: this function is only available when UseLegacySQL=True.

expression: The numeric expression to compute quantiles on.

buckets: The number of buckets.

STDDEV(numeric_expression)

Returns the standard deviation of the values computed by numeric_expr. Rows with a NULL value are not included in the calculation.

numeric_expression: The series to calculate STDDEV on.

STDDEV_POP(numeric_expression)

Computes the population standard deviation of the value computed by numeric_expr.

numeric_expression: The series to calculate STDDEV on.

STDDEV_SAMP(numeric_expression)

Computes the sample standard deviation of the value computed by numeric_expr.

numeric_expression: The series to calculate STDDEV on.

STRING_AGG(expression[, delimiter])

Returns a value (either STRING or BYTES) obtained by concatenating non-null values. If a delimiter is specified, concatenated values are separated by that delimiter; otherwise, a comma is used as a delimiter.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The string expression to concatenate.

delimiter: The delimiter to separate concatenated values.

SUM(DISTINCT expression)

Returns the sum on non-null values. Each distinct value of expression is aggregated only once into the result.

expression: The expression to use to compute the sum.

TOP(column [,max_values][,multiplier])

TOP is a function that is an alternative to the GROUP BY clause. It is used as simplified syntax for GROUP BY ... ORDER BY ... LIMIT .... Generally, the TOP function performs faster than the full ... GROUP BY ... ORDER BY ... LIMIT ... query, but may only return approximate results.
Note: this function is only available when UseLegacySQL=True.

numeric_expression: The series to calculate STDDEV on.

max_values: The maximum number of results to return. Default is 20.

multiplier: A positive integer that increases the value(s) returned by COUNT(*) by the multiple specified.

UNIQUE(expression)

Returns the set of unique, non-NULL values in the scope of the function in an undefined order.
Note: this function is only available when UseLegacySQL=True.

expression: Any expression.

VARIANCE(numeric_expression)

Computes the variance of the values computed by numeric_expr. Rows with a NULL value are not included in the calculation.

numeric_expression: The series to calculate VARIANCE on.

VAR_POP(numeric_expression)

Computes the population variance of the values computed by numeric_expr.

numeric_expression: The series to calculate VARIANCE on.

VAR_SAMP(numeric_expression)

Computes the sample variance of the values computed by numeric_expr.

numeric_expression: The series to calculate VARIANCE on.

CAST(expression)

Cast is used in a query to indicate that the result type of an expression should be converted to some other type.

expression: The expression to cast.

SAFE_CAST(expression)

Cast is used in a query to indicate that the result type of an expression should be converted to some other type.SAFE_CAST is identical to CAST, except it returns NULL instead of raising an error.

expression: The expression to cast.

DATE(timestamp)

Returns a human-readable string of a TIMESTAMP data type in the format %Y-%m-%d.

timestamp: The timestamp from which to return the date.

DATE_ADD(timestamp, interval, interval_units)

Adds the specified interval to a TIMESTAMP data type.

timestamp: The timestamp to add the interval to.

interval: The interval amount to add to the timestamp.

interval_units: The interval unit for interval. Possible values include: YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND.

DATEDIFF(timestamp1, timestamp2)

Returns the number of days between two TIMESTAMP data types. The result is positive if the first TIMESTAMP data type comes after the second TIMESTAMP data type, and otherwise the result is negative.
Note: this function is only available when UseLegacySQL=True.

timestamp1: The first timestamp.

timestamp2: The second timestamp.

DATE_DIFF(timestamp1, timestamp2, date_part)

Computes the number of specified date_part differences between two date expressions. This can be thought of as the number of date_part boundaries crossed between the two dates. If the first date occurs before the second date, then the result is negative.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp1: The first timestamp.

timestamp2: The second timestamp.

date_part: The date part. Supported values are: DAY, MONTH, QUARTER, YEAR.

DAY(timestamp)

Returns the day of the month of a TIMESTAMP data type as an integer between 1 and 31, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the day of the month.

DAYOFWEEK(timestamp)

Returns the day of the week of a TIMESTAMP data type as an integer between 1 (Sunday) and 7 (Saturday), inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the day of the week.

DAYOFYEAR(timestamp)

Returns the day of the year of a TIMESTAMP data type as an integer between 1 and 366, inclusively. The integer 1 refers to January 1.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the day of the year.

FORMAT_UTC_USEC(unix_timestamp)

Returns a human-readable string representation of a UNIX timestamp in the format YYYY-MM-DD HH:MM:SS.uuuuuu.
Note: this function is only available when UseLegacySQL=True.

timestamp: The unix timestamp to format.

HOUR(timestamp)

Returns the hour of a TIMESTAMP data type as an integer between 0 and 23, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the hour as an integer.

MINUTE(timestamp)

Returns the minutes of a TIMESTAMP data type as an integer between 0 and 59, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the minutes as an integer.

MONTH(timestamp)

Returns the month of a TIMESTAMP data type as an integer between 1 and 12, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the month as an integer.

MSEC_TO_TIMESTAMP(unix_timestamp)

Converts a UNIX timestamp in milliseconds to a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

timestamp: The unix timestamp to convert.

PARSE_UTC_USEC(date_string)

Converts a date string to a UNIX timestamp in microseconds. date_string must have the format YYYY-MM-DD HH:MM:SS[.uuuuuu]. The fractional part of the second can be up to 6 digits long or can be omitted.
Note: this function is only available when UseLegacySQL=True.

date_string: The date string to convert.

QUARTER(timestamp)

Returns the quarter of the year of a TIMESTAMP data type as an integer between 1 and 4, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the quarter as an integer.

SEC_TO_TIMESTAMP(unix_timestamp)

Converts a UNIX timestamp in seconds to a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

SECOND(timestamp)

Returns the seconds of a TIMESTAMP data type as an integer between 0 and 59, inclusively. During a leap second, the integer range is between 0 and 60, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the second as an integer.

STRFTIME_UTC_USEC(unix_timestamp, date_format_str)

Returns a human-readable date string in the format date_format_str.date_format_str can include date-related punctuation characters (such as / and -) and special characters accepted by the strftime function in C++ (such as %d for day of month).
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

date_format_str: The date format string.

TIME(timestamp)

Returns a human-readable string of a TIMESTAMP data type, in the format %H:%M:%S.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the time.

TIMESTAMP(date_string)

Convert a date string to a TIMESTAMP data type.

timestamp: The date string to convert.

TIMESTAMP_SECONDS(unix_timestamp)

Interprets INT64_expression as the number of seconds since 1970-01-01 00:00:00 UTC.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The unix timestamp to convert.

TIMESTAMP_MILLIS(unix_timestamp)

Interprets INT64_expression as the number of milliseconds since 1970-01-01 00:00:00 UTC.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The unix timestamp to convert.

TIMESTAMP_MICROS(unix_timestamp)

Interprets INT64_expression as the number of microseconds since 1970-01-01 00:00:00 UTC.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The unix timestamp to convert.

TIMESTAMP_TO_MSEC(timestamp)

Converts a TIMESTAMP data type to a UNIX timestamp in milliseconds.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp to convert.

TIMESTAMP_TO_SEC(timestamp)

Converts a TIMESTAMP data type to a UNIX timestamp in seconds.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp to convert.

TIMESTAMP_TO_USEC(timestamp)

Converts a TIMESTAMP data type to a UNIX timestamp in microseconds.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp to convert.

UNIX_DATE(date_string)

Returns the number of days since 1970-01-01.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

date_string: The date string to convert.

UNIX_SECONDS(timestamp)

Returns the number of seconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The timestamp to convert.

UNIX_MILLIS(timestamp)

Returns the number of milliseconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The timestamp to convert.

UNIX_MICROS(timestamp)

Returns the number of microseconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The timestamp to convert.

USEC_TO_TIMESTAMP(unix_timestamp)

Converts a UNIX timestamp in microseconds to a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

UTC_USEC_TO_DAY(unix_timestamp)

Shifts a UNIX timestamp in microseconds to the beginning of the day it occurs in.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

UTC_USEC_TO_HOUR(unix_timestamp)

Shifts a UNIX timestamp in microseconds to the beginning of the hour it occurs in.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

UTC_USEC_TO_MONTH(unix_timestamp)

Shifts a UNIX timestamp in microseconds to the beginning of the month it occurs in.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

UTC_USEC_TO_WEEK(unix_timestamp, day_of_week)

Returns a UNIX timestamp in microseconds that represents a day in the week of the unix_timestamp argument.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

day_of_week: A day of the week from 0 (Sunday) to 6 (Saturday).

UTC_USEC_TO_YEAR(unix_timestamp)

Returns a UNIX timestamp in microseconds that represents the year of the unix_timestamp argument.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

WEEK(timestamp)

Returns the week of a TIMESTAMP data type as an integer between 1 and 53, inclusively. Weeks begin on Sunday, so if January 1 is on a day other than Sunday, week 1 has fewer than 7 days and the first Sunday of the year is the first day of week 2.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the week as an integer.

YEAR(timestamp)

Returns the year of a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the year as an integer.

ABS(expression)

Returns the absolute value of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

ACOS(expression)

Returns the arc cosine of the argument.

expression: Any column or literal expression.

ACOSH(expression)

Returns the arc hyperbolic cosine of the argument.

expression: Any column or literal expression.

ASIN(expression)

Returns arcsine in radians.

expression: Any column or literal expression.

ASINH(expression)

Returns the arc hyperbolic sine of the argument.

expression: Any column or literal expression.

ATAN(expression)

Returns arc tangent of the argument.

expression: Any column or literal expression.

ATANH(expression)

Returns the arc hyperbolic tangent of the argument.

expression: Any column or literal expression.

ATAN2(expression1, expression2)

Returns the arc tangent of the two arguments.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

CEIL(expression)

Rounds the argument up to the nearest whole number and returns the rounded value.

expression: Any column or literal expression.

COS(expression)

Returns the cosine of the argument.

expression: Any column or literal expression.

COSH(expression)

Returns the hyperbolic cosine of the argument.

expression: Any column or literal expression.

DEGREES(expression)

Returns expression, converted from radians to degrees.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

EXP(expression)

Returns the result of raising the constant "e" - the base of the natural logarithm - to the power of expression.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

FLOOR(expression)

Rounds the argument down to the nearest whole number and returns the rounded value.

expression: Any column or literal expression.

LN(expression)

Returns the natural logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

LOG(expression)

Returns the natural logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

LOG2(expression)

Returns the Base-2 logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

LOG10(expression)

Returns the Base-10 logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

PI()

Returns PI.
Note: this function is only available when UseLegacySQL=True.

POW(expression1, expression2)

Returns the result of raising expression1 to the power of expression2.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

RADIANS(expression)

Returns expression, converted from degrees to radians.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

RAND([expression])

Returns a random float value in the range 0.0 >= value < 1.0. Each int32_seed value always generates the same sequence of random numbers within a given query, as long as you don't use a LIMIT clause. If int32_seed is not specified, BigQuery uses the current timestamp as the seed value.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

ROUND(expression [, integer_digits])

Rounds the argument either up or down to the nearest whole number (or if specified, to the specified number of digits) and returns the rounded value.

expression: Any column or literal expression.

integer_digits: The number of digits to round to.

SIN(expression)

Returns the sine of the argument.

expression: Any column or literal expression.

SINH(expression)

Returns the hyperbolic sine of the argument.

expression: Any column or literal expression.

SQRT(expression)

Returns the square root of the expression.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

TAN(expression)

Returns the tangent of the argument.

expression: Any column or literal expression.

TANH(expression)

Returns the hyperbolic tangent of the argument.

expression: Any column or literal expression.

TRUNC(expression [, integer_digits])

Rounds X to the nearest integer whose absolute value is not greater than Xs. When the integer_digits parameter is specified this function is similar to ROUND(X, N) but always rounds towards zero. Unlike ROUND(X, N) it never overflows.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: Any column or literal expression.

integer_digits: The number of digits to round to.

BYTE_LENGTH(str)

Returns the length of the value in bytes, regardless of whether the type of the value is STRING or BYTES.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to calculate the length on.

CHAR_LENGTH(str)

Returns the length of the STRING in characters.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to calculate the length on.

CONCAT(str1, str2 [, str3] [, ...])

Returns the concatenation of two or more strings, or NULL if any of the values are NULL.

str1: The first string to concatenate.

str2: The second string to concatenate.

str3: The third string to concatenate.

ENDS_WITH(str1, str2)

Takes two values. Returns TRUE if the second value is a suffix of the first.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to search in.

str2: The string to search for.

FROM_BASE64(string_expr)

Converts the base64-encoded input string_expr into BYTES format. To convert BYTES to a base64-encoded STRING, use TO_BASE64.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert from base64 encoding.

FROM_HEX(string_expr)

Converts a hexadecimal-encoded STRING into BYTES format. Returns an error if the input STRING contains characters outside the range (0..9, A..F, a..f). The lettercase of the characters does not matter. To convert BYTES to a hexadecimal-encoded STRING, use TO_HEX.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert from hexadecimal encoding.

INSTR(str1, str2)

Returns the one-based index of the first occurrence of str2 in str1, or returns 0 if str2 does not occur in str1.
Note: this function is only available when UseLegacySQL=True.

str1: The string to search in.

str2: The string to search for.

LEFT(str, numeric_expression)

Returns the leftmost numeric_expr characters of str. If the number is longer than str, the full string will be returned. Example: LEFT('seattle', 3) returns sea.
Note: this function is only available when UseLegacySQL=True.

str: The string to perform the LEFT operation on.

numeric_expression: The number of characters to return.

LENGTH(str)

Returns a numerical value for the length of the string. Example: if str is '123456', LENGTH returns 6.

str: The string to calculate the length on.

LOWER(str)

Returns the original string with all characters in lower case.

str: The string to lower.

LPAD(str1, numeric_expression[, str2])

Pads str1 on the left with str2, repeating str2 until the result string is exactly numeric_expr characters. Example: LPAD('1', 7, '?') returns ??????1.

str1: The string to pad.

numeric_expression: The number of str2 instances to pad.

str2: The pad characters.

LTRIM(str1 [, str2])

Removes characters from the left side of str1. If str2 is omitted, LTRIM removes spaces from the left side of str1. Otherwise, LTRIM removes any characters in str2 from the left side of str1 (case-sensitive).

str1: The string to trim.

str2: The characters to trim from str1.

REPEAT(str, repetitions)

Returns a value that consists of original_value, repeated. The repetitions parameter specifies the number of times to repeat original_value. Returns NULL if either original_value or repetitions are NULL. This function return an error if the repetitions value is negative.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to repeat.

str2: The number of repititions.

REPLACE(str1, str2, str3)

Replaces all instances of str2 within str1 with str3.

str1: The string to search in.

str2: The string to search for.

str3: The string to replace instances of str2.

REVERSE(str)

Returns the reverse of the input STRING or BYTES.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to reverse.

RIGHT(str, numeric_expression)

Returns the rightmost numeric_expr characters of str. If the number is longer than the string, it will return the whole string. Example: RIGHT('kirkland', 4) returns land.
Note: this function is only available when UseLegacySQL=True.

str: The string to perform the RIGHT operation on.

numeric_expression: The number of characters to return.

RPAD(str1, numeric_expression, str2)

Pads str1 on the right with str2, repeating str2 until the result string is exactly numeric_expr characters. Example: RPAD('1', 7, '?') returns 1??????.

str1: The string to pad.

numeric_expression: The number of str2 instances to pad.

str2: The pad characters.

RTRIM(str1 [, str2])

Removes trailing characters from the right side of str1. If str2 is omitted, RTRIM removes trailing spaces from str1. Otherwise, RTRIM removes any characters in str2 from the right side of str1 (case-sensitive).

str1: The string to trim.

str2: The characters to trim from str1.

SPLIT(str [, delimiter])

Splits a string into repeated substrings. If delimiter is specified, the SPLIT function breaks str into substrings, using delimiter as the delimiter.

str: The string to split.

delimiter: The delimiter to split the string on. Default delimiter is a comma (,).

STARTS_WITH(str1, str2)

Takes two values. Returns TRUE if the second value is a prefix of the first.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to search in.

str2: The string to search for.

STRPOS(str1, str2)

Returns the 1-based index of the first occurrence of value2 inside value1. Returns 0 if value2 is not found.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to search in.

str2: The string to search for.

SUBSTR(str, index [, max_len])

Returns a substring of str, starting at index. If the optional max_len parameter is used, the returned string is a maximum of max_len characters long. Counting starts at 1, so the first character in the string is in position 1 (not zero). If index is 5, the substring begins with the 5th character from the left in str. If index is -4, the substring begins with the 4th character from the right in str. Example: SUBSTR('awesome', -4, 4) returns the substring some.

str: The original string.

index: The starting index.

max_len: The maximum length of the return string.

TO_BASE64(string_expr)

Converts a sequence of BYTES into a base64-encoded STRING. To convert a base64-encoded STRING into BYTES, use FROM_BASE64.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert to base64 encoding.

TO_HEX(string_expr)

Converts a sequence of BYTES into a hexadecimal STRING. Converts each byte in the STRING as two hexadecimal characters in the range (0..9, a..f). To convert a hexadecimal-encoded STRING to BYTES, use FROM_HEX.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert to hexadecimal encoding.

TRIM(str1[, str2])

Removes all leading and trailing characters that match value2. If value2 is not specified, all leading and trailing whitespace characters (as defined by the Unicode standard) are removed. If the first argument is of type BYTES, the second argument is required. If value2 contains more than one character or byte, the function removes all leading or trailing characters or bytes contained in value2.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to trim.

str2: The optional string characters to trim from str1.

UPPER(str)

Returns the original string with all characters in upper case.

str: The string to upper.

JSON_EXTRACT(json, json_path)

Selects a value in json according to the JSONPath expression json_path. json_path must be a string constant. Returns the value in JSON string format.

json: The JSON to select a value from.

json_path: The JSON path of the value contained in json.

JSON_EXTRACT_SCALAR(json, json_path)

Selects a value in json according to the JSONPath expression json_path. json_path must be a string constant, and bracket notation is not supported. Returns a scalar JSON value.

json: The JSON to select a value from.

json_path: The JSON path of the value contained in json.

REGEXP_CONTAINS(str, reg_exp)

Returns TRUE if value is a partial match for the regular expression, regex. You can search for a full match by using ^ (beginning of text) and $ (end of text). If the regex argument is invalid, the function returns an error.
Note: this function is only available when UseLegacySQL=True.

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

REGEXP_EXTRACT(str, reg_exp)

Returns the portion of str that matches the capturing group within the regular expression.

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

REGEXP_EXTRACT_ALL(str, reg_exp)

Returns an array of all substrings of value that match the regular expression, regex. The REGEXP_EXTRACT_ALL function only returns non-overlapping matches. For example, using this function to extract ana from banana returns only one substring, not two.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

REGEXP_REPLACE(orig_str, reg_exp, replace_str)

Returns a string where any substring of orig_str that matches reg_exp is replaced with replace_str. For example, REGEXP_REPLACE ('Hello', 'lo', 'p') returns Help.

orig_str: The original string to match in the regular expression.

reg_exp: The regular expression to match.

replace_str: The replacement for the matched orig_str in the regular expression.

FORMAT_IP(integer_value)

Converts 32 least significant bits of integer_value to human-readable IPv4 address string.
Note: this function is only available when UseLegacySQL=True.

integer_value: The integer value to convert to an IPv4 address.

PARSE_IP(readable_ip)

Converts a string representing IPv4 address to unsigned integer value. For example, PARSE_IP('0.0.0.1') will return 1. If string is not a valid IPv4 address, PARSE_IP will return NULL.
Note: this function is only available when UseLegacySQL=True.

readable_ip: The IPv4 address to convert to an integer.

NET.IPV4_FROM_INT64(integer_value)

Converts an IPv4 address from integer format to binary (BYTES) format in network byte order. In the integer input, the least significant bit of the IP address is stored in the least significant bit of the integer, regardless of host or client architecture. For example, 1 means 0.0.0.1, and 0x1FF means 0.0.1.255.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

integer_value: The integer value to convert to an IPv4 address.

NET.IPV4_TO_INT64(readable_ip)

Converts an IPv4 address from binary (BYTES) format in network byte order to integer format. In the integer output, the least significant bit of the IP address is stored in the least significant bit of the integer, regardless of host or client architecture. For example, 1 means 0.0.0.1, and 0x1FF means 0.0.1.255. The output is in the range [0, 0xFFFFFFFF]. If the input length is not 4, this function throws an error. This function does not support IPv6.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

readable_ip: The IPv4 address to convert to an integer.

FARM_FINGERPRINT(expression)

Computes the fingerprint of the STRING or BYTES input using the Fingerprint64 function from the open-source FarmHash library. The output of this function for a particular input will never change.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the fingerprint.

MD5(expression)

Computes the hash of the input using the MD5 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 16 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SHA1(expression)

Computes the hash of the input using the SHA-1 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 20 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SHA256(expression)

Computes the hash of the input using the SHA-256 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 32 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SHA512(expression)

Computes the hash of the input using the SHA-512 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 64 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

JSON_EXTRACT(json, jsonpath)

Selects any value in a JSON array or object. The path to the array is specified in the jsonpath argument. Return value is numeric or null.

json: The JSON document to extract.

jsonpath: The XPath used to select the nodes. The JSONPath must be a string constant. The values of the nodes selected will be returned in a token-separated list.

Predicate Functions

REGEXP_MATCH(str, reg_exp)

Returns true if str matches the regular expression. For string matching without regular expressions, use CONTAINS instead of REGEXP_MATCH.
Note: this function is only available when UseLegacySQL=True.

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

CAST(expression)

Cast is used in a query to indicate that the result type of an expression should be converted to some other type.

expression: The expression to cast.

SAFE_CAST(expression)

Cast is used in a query to indicate that the result type of an expression should be converted to some other type.SAFE_CAST is identical to CAST, except it returns NULL instead of raising an error.

expression: The expression to cast.

DATE(timestamp)

Returns a human-readable string of a TIMESTAMP data type in the format %Y-%m-%d.

timestamp: The timestamp from which to return the date.

DATE_ADD(timestamp, interval, interval_units)

Adds the specified interval to a TIMESTAMP data type.

timestamp: The timestamp to add the interval to.

interval: The interval amount to add to the timestamp.

interval_units: The interval unit for interval. Possible values include: YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND.

DATEDIFF(timestamp1, timestamp2)

Returns the number of days between two TIMESTAMP data types. The result is positive if the first TIMESTAMP data type comes after the second TIMESTAMP data type, and otherwise the result is negative.
Note: this function is only available when UseLegacySQL=True.

timestamp1: The first timestamp.

timestamp2: The second timestamp.

DATE_DIFF(timestamp1, timestamp2, date_part)

Computes the number of specified date_part differences between two date expressions. This can be thought of as the number of date_part boundaries crossed between the two dates. If the first date occurs before the second date, then the result is negative.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp1: The first timestamp.

timestamp2: The second timestamp.

date_part: The date part. Supported values are: DAY, MONTH, QUARTER, YEAR.

DAY(timestamp)

Returns the day of the month of a TIMESTAMP data type as an integer between 1 and 31, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the day of the month.

DAYOFWEEK(timestamp)

Returns the day of the week of a TIMESTAMP data type as an integer between 1 (Sunday) and 7 (Saturday), inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the day of the week.

DAYOFYEAR(timestamp)

Returns the day of the year of a TIMESTAMP data type as an integer between 1 and 366, inclusively. The integer 1 refers to January 1.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the day of the year.

FORMAT_UTC_USEC(unix_timestamp)

Returns a human-readable string representation of a UNIX timestamp in the format YYYY-MM-DD HH:MM:SS.uuuuuu.
Note: this function is only available when UseLegacySQL=True.

timestamp: The unix timestamp to format.

HOUR(timestamp)

Returns the hour of a TIMESTAMP data type as an integer between 0 and 23, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the hour as an integer.

MINUTE(timestamp)

Returns the minutes of a TIMESTAMP data type as an integer between 0 and 59, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the minutes as an integer.

MONTH(timestamp)

Returns the month of a TIMESTAMP data type as an integer between 1 and 12, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the month as an integer.

MSEC_TO_TIMESTAMP(unix_timestamp)

Converts a UNIX timestamp in milliseconds to a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

timestamp: The unix timestamp to convert.

PARSE_UTC_USEC(date_string)

Converts a date string to a UNIX timestamp in microseconds. date_string must have the format YYYY-MM-DD HH:MM:SS[.uuuuuu]. The fractional part of the second can be up to 6 digits long or can be omitted.
Note: this function is only available when UseLegacySQL=True.

date_string: The date string to convert.

QUARTER(timestamp)

Returns the quarter of the year of a TIMESTAMP data type as an integer between 1 and 4, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the quarter as an integer.

SEC_TO_TIMESTAMP(unix_timestamp)

Converts a UNIX timestamp in seconds to a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

SECOND(timestamp)

Returns the seconds of a TIMESTAMP data type as an integer between 0 and 59, inclusively. During a leap second, the integer range is between 0 and 60, inclusively.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the second as an integer.

STRFTIME_UTC_USEC(unix_timestamp, date_format_str)

Returns a human-readable date string in the format date_format_str.date_format_str can include date-related punctuation characters (such as / and -) and special characters accepted by the strftime function in C++ (such as %d for day of month).
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

date_format_str: The date format string.

TIME(timestamp)

Returns a human-readable string of a TIMESTAMP data type, in the format %H:%M:%S.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the time.

TIMESTAMP(date_string)

Convert a date string to a TIMESTAMP data type.

timestamp: The date string to convert.

TIMESTAMP_SECONDS(unix_timestamp)

Interprets INT64_expression as the number of seconds since 1970-01-01 00:00:00 UTC.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The unix timestamp to convert.

TIMESTAMP_MILLIS(unix_timestamp)

Interprets INT64_expression as the number of milliseconds since 1970-01-01 00:00:00 UTC.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The unix timestamp to convert.

TIMESTAMP_MICROS(unix_timestamp)

Interprets INT64_expression as the number of microseconds since 1970-01-01 00:00:00 UTC.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The unix timestamp to convert.

TIMESTAMP_TO_MSEC(timestamp)

Converts a TIMESTAMP data type to a UNIX timestamp in milliseconds.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp to convert.

TIMESTAMP_TO_SEC(timestamp)

Converts a TIMESTAMP data type to a UNIX timestamp in seconds.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp to convert.

TIMESTAMP_TO_USEC(timestamp)

Converts a TIMESTAMP data type to a UNIX timestamp in microseconds.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp to convert.

UNIX_DATE(date_string)

Returns the number of days since 1970-01-01.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

date_string: The date string to convert.

UNIX_SECONDS(timestamp)

Returns the number of seconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The timestamp to convert.

UNIX_MILLIS(timestamp)

Returns the number of milliseconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The timestamp to convert.

UNIX_MICROS(timestamp)

Returns the number of microseconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

timestamp: The timestamp to convert.

USEC_TO_TIMESTAMP(unix_timestamp)

Converts a UNIX timestamp in microseconds to a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

UTC_USEC_TO_DAY(unix_timestamp)

Shifts a UNIX timestamp in microseconds to the beginning of the day it occurs in.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

UTC_USEC_TO_HOUR(unix_timestamp)

Shifts a UNIX timestamp in microseconds to the beginning of the hour it occurs in.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

UTC_USEC_TO_MONTH(unix_timestamp)

Shifts a UNIX timestamp in microseconds to the beginning of the month it occurs in.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

UTC_USEC_TO_WEEK(unix_timestamp, day_of_week)

Returns a UNIX timestamp in microseconds that represents a day in the week of the unix_timestamp argument.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to shift.

day_of_week: A day of the week from 0 (Sunday) to 6 (Saturday).

UTC_USEC_TO_YEAR(unix_timestamp)

Returns a UNIX timestamp in microseconds that represents the year of the unix_timestamp argument.
Note: this function is only available when UseLegacySQL=True.

unix_timestamp: The unix timestamp to convert.

WEEK(timestamp)

Returns the week of a TIMESTAMP data type as an integer between 1 and 53, inclusively. Weeks begin on Sunday, so if January 1 is on a day other than Sunday, week 1 has fewer than 7 days and the first Sunday of the year is the first day of week 2.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the week as an integer.

YEAR(timestamp)

Returns the year of a TIMESTAMP data type.
Note: this function is only available when UseLegacySQL=True.

timestamp: The timestamp from which to return the year as an integer.

ABS(expression)

Returns the absolute value of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

ACOS(expression)

Returns the arc cosine of the argument.

expression: Any column or literal expression.

ACOSH(expression)

Returns the arc hyperbolic cosine of the argument.

expression: Any column or literal expression.

ASIN(expression)

Returns arcsine in radians.

expression: Any column or literal expression.

ASINH(expression)

Returns the arc hyperbolic sine of the argument.

expression: Any column or literal expression.

ATAN(expression)

Returns arc tangent of the argument.

expression: Any column or literal expression.

ATANH(expression)

Returns the arc hyperbolic tangent of the argument.

expression: Any column or literal expression.

ATAN2(expression1, expression2)

Returns the arc tangent of the two arguments.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

CEIL(expression)

Rounds the argument up to the nearest whole number and returns the rounded value.

expression: Any column or literal expression.

COS(expression)

Returns the cosine of the argument.

expression: Any column or literal expression.

COSH(expression)

Returns the hyperbolic cosine of the argument.

expression: Any column or literal expression.

DEGREES(expression)

Returns expression, converted from radians to degrees.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

EXP(expression)

Returns the result of raising the constant "e" - the base of the natural logarithm - to the power of expression.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

FLOOR(expression)

Rounds the argument down to the nearest whole number and returns the rounded value.

expression: Any column or literal expression.

LN(expression)

Returns the natural logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

LOG(expression)

Returns the natural logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

LOG2(expression)

Returns the Base-2 logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

LOG10(expression)

Returns the Base-10 logarithm of the argument.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

PI()

Returns PI.
Note: this function is only available when UseLegacySQL=True.

POW(expression1, expression2)

Returns the result of raising expression1 to the power of expression2.

expression1: Any column or literal expression.

expression2: Any column or literal expression.

RADIANS(expression)

Returns expression, converted from degrees to radians.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

RAND([expression])

Returns a random float value in the range 0.0 >= value < 1.0. Each int32_seed value always generates the same sequence of random numbers within a given query, as long as you don't use a LIMIT clause. If int32_seed is not specified, BigQuery uses the current timestamp as the seed value.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

ROUND(expression [, integer_digits])

Rounds the argument either up or down to the nearest whole number (or if specified, to the specified number of digits) and returns the rounded value.

expression: Any column or literal expression.

integer_digits: The number of digits to round to.

SIN(expression)

Returns the sine of the argument.

expression: Any column or literal expression.

SINH(expression)

Returns the hyperbolic sine of the argument.

expression: Any column or literal expression.

SQRT(expression)

Returns the square root of the expression.
Note: this function is only available when UseLegacySQL=True.

expression: Any column or literal expression.

TAN(expression)

Returns the tangent of the argument.

expression: Any column or literal expression.

TANH(expression)

Returns the hyperbolic tangent of the argument.

expression: Any column or literal expression.

TRUNC(expression [, integer_digits])

Rounds X to the nearest integer whose absolute value is not greater than Xs. When the integer_digits parameter is specified this function is similar to ROUND(X, N) but always rounds towards zero. Unlike ROUND(X, N) it never overflows.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: Any column or literal expression.

integer_digits: The number of digits to round to.

BYTE_LENGTH(str)

Returns the length of the value in bytes, regardless of whether the type of the value is STRING or BYTES.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to calculate the length on.

CHAR_LENGTH(str)

Returns the length of the STRING in characters.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to calculate the length on.

CONCAT(str1, str2 [, str3] [, ...])

Returns the concatenation of two or more strings, or NULL if any of the values are NULL.

str1: The first string to concatenate.

str2: The second string to concatenate.

str3: The third string to concatenate.

ENDS_WITH(str1, str2)

Takes two values. Returns TRUE if the second value is a suffix of the first.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to search in.

str2: The string to search for.

FROM_BASE64(string_expr)

Converts the base64-encoded input string_expr into BYTES format. To convert BYTES to a base64-encoded STRING, use TO_BASE64.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert from base64 encoding.

FROM_HEX(string_expr)

Converts a hexadecimal-encoded STRING into BYTES format. Returns an error if the input STRING contains characters outside the range (0..9, A..F, a..f). The lettercase of the characters does not matter. To convert BYTES to a hexadecimal-encoded STRING, use TO_HEX.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert from hexadecimal encoding.

INSTR(str1, str2)

Returns the one-based index of the first occurrence of str2 in str1, or returns 0 if str2 does not occur in str1.
Note: this function is only available when UseLegacySQL=True.

str1: The string to search in.

str2: The string to search for.

LEFT(str, numeric_expression)

Returns the leftmost numeric_expr characters of str. If the number is longer than str, the full string will be returned. Example: LEFT('seattle', 3) returns sea.
Note: this function is only available when UseLegacySQL=True.

str: The string to perform the LEFT operation on.

numeric_expression: The number of characters to return.

LENGTH(str)

Returns a numerical value for the length of the string. Example: if str is '123456', LENGTH returns 6.

str: The string to calculate the length on.

LOWER(str)

Returns the original string with all characters in lower case.

str: The string to lower.

LPAD(str1, numeric_expression[, str2])

Pads str1 on the left with str2, repeating str2 until the result string is exactly numeric_expr characters. Example: LPAD('1', 7, '?') returns ??????1.

str1: The string to pad.

numeric_expression: The number of str2 instances to pad.

str2: The pad characters.

LTRIM(str1 [, str2])

Removes characters from the left side of str1. If str2 is omitted, LTRIM removes spaces from the left side of str1. Otherwise, LTRIM removes any characters in str2 from the left side of str1 (case-sensitive).

str1: The string to trim.

str2: The characters to trim from str1.

REPEAT(str, repetitions)

Returns a value that consists of original_value, repeated. The repetitions parameter specifies the number of times to repeat original_value. Returns NULL if either original_value or repetitions are NULL. This function return an error if the repetitions value is negative.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to repeat.

str2: The number of repititions.

REPLACE(str1, str2, str3)

Replaces all instances of str2 within str1 with str3.

str1: The string to search in.

str2: The string to search for.

str3: The string to replace instances of str2.

REVERSE(str)

Returns the reverse of the input STRING or BYTES.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to reverse.

RIGHT(str, numeric_expression)

Returns the rightmost numeric_expr characters of str. If the number is longer than the string, it will return the whole string. Example: RIGHT('kirkland', 4) returns land.
Note: this function is only available when UseLegacySQL=True.

str: The string to perform the RIGHT operation on.

numeric_expression: The number of characters to return.

RPAD(str1, numeric_expression, str2)

Pads str1 on the right with str2, repeating str2 until the result string is exactly numeric_expr characters. Example: RPAD('1', 7, '?') returns 1??????.

str1: The string to pad.

numeric_expression: The number of str2 instances to pad.

str2: The pad characters.

RTRIM(str1 [, str2])

Removes trailing characters from the right side of str1. If str2 is omitted, RTRIM removes trailing spaces from str1. Otherwise, RTRIM removes any characters in str2 from the right side of str1 (case-sensitive).

str1: The string to trim.

str2: The characters to trim from str1.

SPLIT(str [, delimiter])

Splits a string into repeated substrings. If delimiter is specified, the SPLIT function breaks str into substrings, using delimiter as the delimiter.

str: The string to split.

delimiter: The delimiter to split the string on. Default delimiter is a comma (,).

STARTS_WITH(str1, str2)

Takes two values. Returns TRUE if the second value is a prefix of the first.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to search in.

str2: The string to search for.

STRPOS(str1, str2)

Returns the 1-based index of the first occurrence of value2 inside value1. Returns 0 if value2 is not found.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to search in.

str2: The string to search for.

SUBSTR(str, index [, max_len])

Returns a substring of str, starting at index. If the optional max_len parameter is used, the returned string is a maximum of max_len characters long. Counting starts at 1, so the first character in the string is in position 1 (not zero). If index is 5, the substring begins with the 5th character from the left in str. If index is -4, the substring begins with the 4th character from the right in str. Example: SUBSTR('awesome', -4, 4) returns the substring some.

str: The original string.

index: The starting index.

max_len: The maximum length of the return string.

TO_BASE64(string_expr)

Converts a sequence of BYTES into a base64-encoded STRING. To convert a base64-encoded STRING into BYTES, use FROM_BASE64.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert to base64 encoding.

TO_HEX(string_expr)

Converts a sequence of BYTES into a hexadecimal STRING. Converts each byte in the STRING as two hexadecimal characters in the range (0..9, a..f). To convert a hexadecimal-encoded STRING to BYTES, use FROM_HEX.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

string_expr: The string to convert to hexadecimal encoding.

TRIM(str1[, str2])

Removes all leading and trailing characters that match value2. If value2 is not specified, all leading and trailing whitespace characters (as defined by the Unicode standard) are removed. If the first argument is of type BYTES, the second argument is required. If value2 contains more than one character or byte, the function removes all leading or trailing characters or bytes contained in value2.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str1: The string to trim.

str2: The optional string characters to trim from str1.

UPPER(str)

Returns the original string with all characters in upper case.

str: The string to upper.

JSON_EXTRACT(json, json_path)

Selects a value in json according to the JSONPath expression json_path. json_path must be a string constant. Returns the value in JSON string format.

json: The JSON to select a value from.

json_path: The JSON path of the value contained in json.

JSON_EXTRACT_SCALAR(json, json_path)

Selects a value in json according to the JSONPath expression json_path. json_path must be a string constant, and bracket notation is not supported. Returns a scalar JSON value.

json: The JSON to select a value from.

json_path: The JSON path of the value contained in json.

REGEXP_CONTAINS(str, reg_exp)

Returns TRUE if value is a partial match for the regular expression, regex. You can search for a full match by using ^ (beginning of text) and $ (end of text). If the regex argument is invalid, the function returns an error.
Note: this function is only available when UseLegacySQL=True.

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

REGEXP_EXTRACT(str, reg_exp)

Returns the portion of str that matches the capturing group within the regular expression.

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

REGEXP_EXTRACT_ALL(str, reg_exp)

Returns an array of all substrings of value that match the regular expression, regex. The REGEXP_EXTRACT_ALL function only returns non-overlapping matches. For example, using this function to extract ana from banana returns only one substring, not two.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

str: The string to match in the regular expression.

reg_exp: The regular expression to match.

REGEXP_REPLACE(orig_str, reg_exp, replace_str)

Returns a string where any substring of orig_str that matches reg_exp is replaced with replace_str. For example, REGEXP_REPLACE ('Hello', 'lo', 'p') returns Help.

orig_str: The original string to match in the regular expression.

reg_exp: The regular expression to match.

replace_str: The replacement for the matched orig_str in the regular expression.

FORMAT_IP(integer_value)

Converts 32 least significant bits of integer_value to human-readable IPv4 address string.
Note: this function is only available when UseLegacySQL=True.

integer_value: The integer value to convert to an IPv4 address.

PARSE_IP(readable_ip)

Converts a string representing IPv4 address to unsigned integer value. For example, PARSE_IP('0.0.0.1') will return 1. If string is not a valid IPv4 address, PARSE_IP will return NULL.
Note: this function is only available when UseLegacySQL=True.

readable_ip: The IPv4 address to convert to an integer.

NET.IPV4_FROM_INT64(integer_value)

Converts an IPv4 address from integer format to binary (BYTES) format in network byte order. In the integer input, the least significant bit of the IP address is stored in the least significant bit of the integer, regardless of host or client architecture. For example, 1 means 0.0.0.1, and 0x1FF means 0.0.1.255.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

integer_value: The integer value to convert to an IPv4 address.

NET.IPV4_TO_INT64(readable_ip)

Converts an IPv4 address from binary (BYTES) format in network byte order to integer format. In the integer output, the least significant bit of the IP address is stored in the least significant bit of the integer, regardless of host or client architecture. For example, 1 means 0.0.0.1, and 0x1FF means 0.0.1.255. The output is in the range [0, 0xFFFFFFFF]. If the input length is not 4, this function throws an error. This function does not support IPv6.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

readable_ip: The IPv4 address to convert to an integer.

FARM_FINGERPRINT(expression)

Computes the fingerprint of the STRING or BYTES input using the Fingerprint64 function from the open-source FarmHash library. The output of this function for a particular input will never change.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the fingerprint.

MD5(expression)

Computes the hash of the input using the MD5 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 16 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SHA1(expression)

Computes the hash of the input using the SHA-1 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 20 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SHA256(expression)

Computes the hash of the input using the SHA-256 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 32 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SHA512(expression)

Computes the hash of the input using the SHA-512 algorithm. The input can either be STRING or BYTES. The string version treats the input as an array of bytes. This function returns 64 bytes.
Note: this function is only available when using Standard SQL (UseLegacySQL=False).

expression: The expression to use to compute the hash.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT actor.attributes.email, repository.name INTO
"csv://publicdata:samples.github_nested.txt" FROM
"publicdata:samples.github_nested" WHERE repository.name =
'EntityFramework'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT actor.attributes.email, repository.name INTO
"csv://publicdata:samples.github_nested.txt;delimiter=tab" FROM
"publicdata:samples.github_nested" WHERE repository.name =
'EntityFramework'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO publicdata:samples.github_nested
(repository.name) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "CoreCLR");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
```

```
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV Google Sheets Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to a Spreadsheet

You can connect to a spreadsheet by providing authentication to Google and then setting the Spreadsheet property to the name of the spreadsheet.

If you want to view a list of information about the spreadsheets in your Google Drive, execute a query to the Spreadsheets view after you authenticate.

Authenticating to Google

You can authenticate with a Google account, a Google Apps account, or a service account. A service account is required to delegate domain-wide access.

The authentication process follows the OAuth 2.0 authentication standard.

Fine-Tuning Data Access

You can use the following connection properties to customize data modeling to reflect how your worksheets are organized.

guides on executing SQL to worksheets and ranges:

Spreadsheet: Set this to the name of the spreadsheet.

If you want to view a list of information about the spreadsheets in your Google Drive, execute a query to the Spreadsheets view.

Header: Set this to true to detect column names if your worksheet has column headers. Set this to false to use A, B, C, etc. The Header property affects inserts and updates as well as queries.

ValueInputOption: By default, this is set to UserEntered and input is parsed exactly as if it were entered into the Google Sheets UI. For example, strings may be converted to numbers, dates, etc.

Set this to Raw to insert input as a string. For example, when ValueInputOption is set to Raw, the insert below places "=1+2" into the cell as a string, not a formula:

```
INSERT INTO Spreadsheet1_Sheet1(Name, Amount) VALUES ('Tom','=1+2'
)
```

ValueRenderOption: Set this to determine how values will be calculated and formatted.

Executing SQL to Spreadsheets and Ranges

See the following for example connection strings and SQL:

Selecting GoogleSheets Data

Inserting GoogleSheets Data

Updating GoogleSheets Data

Deleting GoogleSheets Data

Advanced Tab

Connection String Options

The connection string properties describe the various options that can be used to establish a connection.

The following is the full list of the options you can configure in the connection string for this provider.

Date Time Render Option	Determines how dates, times, and durations should be represented in the output. This is ignored if the ValueRenderOption is FormattedValue. The default datetime render option is SerialNumber.
Define Tables	Define the tables within the Google Spreadsheet.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.

Firewall User	The user name to use to authenticate with a proxy-based firewall.
Header	Indicates whether or not the first row should be used as a column header.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Insert Data Option	Determines how existing data is changed when new data is input.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth JWT Cert	The JWT Certificate store.
OAuth JWT Cert Password	The password for the OAuth JWT certificate.
OAuth JWT Cert Subject	The subject of the OAuth JWT certificate.
OAuth JWT Cert Type	The type of key store containing the JWT Certificate.
OAuth JWT Issuer	The issuer of the Java Web Token.
OAuth JWT Subject	The user subject for which the application is requesting delegated access.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Orientation	Indicates whether the data in the sheet is laid out horizontally or vertically.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per page of data retrieved from the Google Sheets API.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Google Spreadsheets from the provider.
Row Scan Depth	Set this property to control the number of rows scanned when TypeDetectionScheme is set to RowScan.
Show Empty Rows	Indicates whether or not the empty rows should be pushed.
Spreadsheet	The name or Id of the spreadsheet to be viewed.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Type Detection Scheme	Determines how to determine the data types of columns.

Use Simple Names	Boolean determining if simple names should be used for tables and columns.
Value Input Option	Determines how values should be rendered in the output.
Value Render Option	Determines how values should be rendered in the output.

Date Time Render Option

Determines how dates, times, and durations should be represented in the output. This is ignored if the ValueRenderOption is FormattedValue. The default datetime render option is SerialNumber.

Data Type

string

Default Value

"SerialNumber"

Remarks

SerialNumber	Instructs the adapter to output date, time, datetime, and duration fields as doubles in "serial number" format, as popularized by Lotus 1-2-3. The whole number portion of the value (left of the decimal) counts the days since December 30th 1899. The fractional portion (right of the decimal) counts the time as a fraction of the day. For example, January 1st 1900 at noon would be 2.5, 2 because it's 2 days after December 30st, 1899, and .5 because noon is half a day. February 1st, 1900 at 3pm would be 33.625. This correctly treats the year 1900 as not a leap year.
FormattedString	Instructs the adapter to output date, time, datetime, and duration fields as strings in their given number format (which is dependent on the spreadsheet locale).

Define Tables

Define the tables within the Google Spreadsheet.

Data Type

string

Default Value

""

Remarks

This property is used to define the ranges within a sheet that will appear as tables. The value is a comma-separated list of name-value pairs in the form [Table Name]=[Spreadsheet Name]_[Sheet Name]![Range] or [Table Name]=[Spreadsheet Name]_[Sheet Name]![Range]. Table Name is the name of the table you want to use for the data and will be used when issuing queries. Sheet Name is the name of the sheet within the Google Spreadsheet and Range is the range of cells that contain the data for the table.

Here is an example DefineTables value: DefineTables="Table1=Spreadsheet1_Sheet1!A1:N25,Table2=Spreadsheet1_Sheet2!C3:M53,Table4=xIsPcLs2-bF3AavQcSLCfzs3kGc_Sheet4!C20:N60".

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description

TUNNEL	80	When this is set, the adapter opens a connection to Google Sheets and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Header

Indicates whether or not the first row should be used as a column header.

Data Type

bool

Default Value

true

Remarks

If true, the first row will be used as a column header. Otherwise, the pseudo column names (A, B, C, etc.) will be used.

The Header property is used in conjunction with the Orientation property. When Header is set to false and Orientation is set to Columns, column names are reported as R1, R2, R3, etc.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Insert Data Option

Determines how existing data is changed when new data is input.

Data Type

string

Default Value

"Overwrite"

Remarks

Overwrite	The new data overwrites existing data in the areas it is written.
InsertRows	Rows are inserted for the new data.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth JWT Cert

The JWT Certificate store.

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The OAuthJWTCertType field specifies the type of the certificate store specified by OAuthJWTCert. If the store is password protected, specify the password in OAuthJWTCertPassword.

OAuthJWTCert is used in conjunction with the OAuthJWTCertSubject field in order to specify client certificates. If OAuthJWTCert has a value, and OAuthJWTCertSubject is set, a search for a certificate is initiated. Please refer to the OAuthJWTCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

OAuth JWT Cert Password

The password for the OAuth JWT certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

OAuth JWT Cert Subject

The subject of the OAuth JWT certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality

S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

OAuth JWT Cert Type

The type of key store containing the JWT Certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.

JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

OAuth JWT Issuer

The issuer of the Java Web Token.

Data Type

string

Default Value

""

Remarks

The issuer of the Java Web Token. This is typically either the Client ID or Email Address of the OAuth Application.

OAuth JWT Subject

The user subject for which the application is requesting delegated access.

Data Type

string

Default Value

""

Remarks

The user subject for which the application is requesting delegated access. Typically, the user account name or email address.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\GoogleSheets Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Orientation

Indicates whether the data in the sheet is laid out horizontally or vertically.

Data Type

string

Default Value

"Vertical"

Remarks

Horizontal	Specifies that the adapter operates on the rows of a sheet.
Vertical	Specifies that the adapter operates on the columns of a sheet.

By default, the adapter models vertically oriented spreadsheet data -- rows arranged vertically below a header row.

Set this to "Horizontal" if the rows are arranged left to right. The first column contains the column names and subsequent columns become rows.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per page of data retrieved from the Google Sheets API.

Data Type

string

Default Value

"1000"

Remarks

The number of results to return per page of data retrieved from the Google Sheets API.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Google Spreadsheets from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Row Scan Depth

Set this property to control the number of rows scanned when TypeDetectionScheme is set to RowScan.

Data Type

string

Default Value

"50"

Remarks

Determines the number of rows to scan to heuristically determine the column data types.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

TypeDetectionScheme must be set to RowScan to enable this property.

Show Empty Rows

Indicates whether or not the empty rows should be pushed.

Data Type

bool

Default Value

false

Remarks

If true, the empty rows will be pushed at the output.

Spreadsheet

The name or Id of the spreadsheet to be viewed.

Data Type

string

Default Value

""

Remarks

The name or Id of the spreadsheet to be viewed. Query the Spreadsheets view to retrieve this data.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIh v.....Qw== -----END CERTIFICATE-----

A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Type Detection Scheme

Determines how to determine the data types of columns.

Data Type

string

Default Value

"RowScan"

Remarks

None	Setting TypeDetectionScheme to None will return all columns as the string type. *Note*: Even when set to None, the column names will still be scanned when Header is set to True.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned.

Use Simple Names

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

Boolean determining if simple names should be used for tables and columns. Google Sheets tables and columns can use special characters in names that are normally not allowed in standard databases. UseSimpleNames makes the adapter easier to use with traditional database tools.

Setting UseSimpleNames to true will simplify the names of tables and columns returned. It will enforce a naming scheme such that only alphanumeric characters and the underscore are valid for the displayed table and column names. Any nonalphanumeric characters will be converted to an underscore.

Value Input Option

Determines how values should be rendered in the output.

Data Type

string

Default Value

"UserEntered"

Remarks

Raw	The values the user has entered will not be parsed and will be stored as-is.
UserEntered	The values will be parsed as if the user typed them into the UI. Numbers will stay as numbers, but strings may be converted to numbers, dates, etc. -- following the same rules that are applied when entering text into a cell via the Google Sheets UI.

Value Render Option

Determines how values should be rendered in the output.

Data Type

string

Default Value

"FormattedValue"

Remarks

FormattedValue	Values will be calculated and formatted in the reply according to the cell's formatting. Formatting is based on the spreadsheet's locale, not the requesting user's locale. For example, if A1 is "1.23" and A2 is "=A1" and formatted as currency, then A2 would return "$1.23".
UnformattedValue	Values will be calculated, but not formatted in the reply. For example, if A1 is "1.23" and A2 is "=A1" and formatted as currency, then A2 would return the number "1.23".
Formula	Values will not be calculated. The reply will include the formulas. For example, if A1 is "1.23" and A2 is "=A1" and formatted as currency, then A2 would return "=A1".

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Google Sheets Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to Google Spreadsheets

This section documents how to provide authentication for your account type and then connect to a spreadsheet.

Authenticate a Google or Google Apps Account

If you have a Google or Google Apps account, you need to use OAuth authentication. See our Using OAuth guide for more information.

Authenticate Using Two-Step Verification

Accounts that use two-step verification also use OAuth. The difference will be that when the adapter opens the Google login page in your default web browser, it will also prompt you to enter your verification code, along with your username and password.

View Information on the Available Spreadsheets

You can query the Spreadsheets view to find a list of names for spreadsheets your account has access to. You can also query this view to access other information about the available spreadsheets: the most recently visited spreadsheet and author information.

Using OAuth

OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Configuring the Adapter

The adapter can be configured for OAuth in three ways using the InitiateOAuth property.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and Google Sheets.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from Google Sheets and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If Google Sheets issues a long-lived access token, use the Google Sheets developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

Advanced Settings

Register Your Application

This step is not typically necessary. The adapter is already registered with Google as an OAuth application. Register your own application if you want to display your own information, instead of adapter information, when users log into Google Sheets to grant permissions to the adapter.

Log into the Google API Console.

Click Create Project or select an existing project.

In the API Manager, click Credentials -> Create Credentials -> OAuth Client Id.

If you are connecting from a desktop application, select Other.

If you are connecting from a Web application, select Web Application. In the Authorized Redirect URIs box, enter the URL you want to be used as a trusted redirect URL, where the user will return with the token that verifies that they have granted your app access. The redirect URL is also called the callback URL.

Click Create. The OAuthClientId and OAuthClientSecret are displayed.

Click Library -> Drive API -> Enable API.

After you create your application, you are shown your OAuth credentials.

OAuthClientId: Set this to the client Id in your app settings.

OAuthClientSecret set this to the client secret in your app settings.

The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

HTTP Proxies

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

Selecting GoogleSheets Data

Google Sheets worksheets can be organized in different ways. The adapter offers settings to select data based on various organizational methods.

With Header Cells

If a worksheet has column headers the adapter can use them to represent the column names of the worksheet table. To use header cells set the Header property to True. Then you can query like so:

```
SELECT Id, Name, Item, Quantity, Amount FROM Spreatsheet1_Sheet1
WHERE Amount = 50
```

The query above assumes that the first row of the spreadsheet has the column names Id, Name, Quantity, and Amount. The spreadsheet may have more columns than were selected and they can be in any order.

Without Header Cells

If a spreadsheet does not have header columns, or if you would like to ignore the header columns, you can set the Header property to false. In this case each column is represented by the column name in Google Spreadsheets; i.e., A, B, C, ...

```
SELECT A, B, C, D FROM Spreatsheet1_Sheet1 WHERE D = 50
```

Orientation

If a spreadsheet is laid out horizontally (i.e., the column names are arranged vertically in the first column) then you can set the Orientation property to Horizontal to read each column as a row in the table. The Orientation property can be used in conjunction with the Header connection property, and the range syntax described below.

If header columns are not set and the dimension is horizontal, the column names will be R1, R2, R3, ...

```
SELECT R1, R2, R3 FROM Spreatsheet1_Sheet1 WHERE R2 = 50
```

Range

You can use the range feature to select data from a specific portion of the worksheet; for example, if you have a sparse spreadsheet that is not top-left aligned or if the data of interest is somewhere in the middle of the sheet. The range is specified as part of the table name. For example the following command will select the range of cells between A1 and E5:

```
SELECT * FROM Spreatsheet1_Sheet#A1:E5
```

Inserting GoogleSheets Data

Insert Data

You can govern how the adapter sets column names by setting the Header connection string property. This affects the commands that are available when inserting data.

Insert with Headers

Header property is set to true.

```
INSERT INTO Spreadsheet1_Sheet1(Name, Amount) VALUES ('Test', 10)
```

Insert without Headers

Header property is set to false.

```
INSERT INTO Spreadsheet1_Sheet1(A, B) VALUES ('Test', 10)
```

Insert with Range

Insert with range is not supported.

Insert or Overwrite

You can choose if you want to overwrite existing data after a table or insert new rows for the new data. By default, the input overwrites data after the table. To write the new data into new rows, set InsertDataOption to InsertRows and specify an Id.

InsertDataOption property is set to Overwrite which is the default.

```
INSERT INTO Spreadsheet1_Sheet1(Name, Amount) VALUES ('Test', 10)
```

InsertDataOption property is set to InsertRow will overwrite data at the end of the table.

```
INSERT INTO Spreadsheet1_Sheet1(Id,Name, Amount) VALUES (7,'Test',
10)
```

Insert without Parsing Input

ValueInputOption property is set to Raw. The input is not parsed and is simply inserted as a string, so the input "=1+2" places the string "=1+2" in the cell, not a formula.

Insert with Parsed Input

ValueInputOption property is set to UserEntered. The input is parsed exactly as if it were entered into the Google Sheets UI.

Updating GoogleSheets Data

Update with headers

When the Header property is set to true, you can execute an UPDATE by specifying the detected column name and the unique row identifier the adapter generates:

```
UPDATE Customers SET Col1='value' WHERE Id=7
```

Update without Headers

When the Header is set to false, you can specify the columns to update explicitly:

```
UPDATE Customers SET H='value' WHERE Id=7
```

Range Notation

Set the columns corresponding to the range and specify the row number as the Id:

```
UPDATE "Customers#A15:C15" SET A='Ana Trujilo', B='Northwind,
Inc.', C='100,000' WHERE Id='15'
```

Deleting GoogleSheets Data

Delete Data

When deleting an entire row of data, you will need to be sure to refresh the table data. This is because Ids are assigned based on the row on the spreadsheet. For example, if there are five rows, with Ids 1 through 5, and you delete row 3 then row 4 will become 3 and 5 will become 4.

Working with Formulas

Formulas require special treatment in Google Sheets documents.

Selecting Formulas

When selecting formulas the adapter can return either the result of the formula or the formula itself:

You can control whether or not the formula is returned using the ValueRenderOption connection property.

Due to limitations of the Google Spreadsheets API, any SELECT executed when the connection string property Header is set to true will return only the value of the formula.

Updating Formulas

The adapter allows you to update cells using formulas. When the ValueInputOption property is set to UserEntered, any value entered beginning with "=" will be treated as a formula. For example, the following command will insert a formula into row 6 column B that will be the result of the sum of cells B1 to B5:

```
UPDATE Spreatsheet1_Sheet1 SET A='Bill', B='=SUM(B1:B5)' WHERE Id=6
```

SQL Compliance

The Google Sheets Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Using Spreadsheets as Tables for information on the capabilities of the Google Sheets API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Google Sheets adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Spreadsheet1_Sheet1
```

Rename a column:

```
SELECT "Column1" AS MY_Column1 FROM Spreadsheet1_Sheet1
```

Search data:

```
SELECT * FROM Spreadsheet1_Sheet1 WHERE Column2 = 'Bob';
```

The Google Sheets APIs support the following operators in the WHERE clause: =.

```
SELECT * FROM Spreadsheet1_Sheet1 WHERE Column2 = 'Bob';
```

Sort a result set in ascending order:

```
SELECT Id, Column1 FROM Spreadsheet1_Sheet1  ORDER BY Column1 ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Column1 INTO "csv://Spreadsheet1_Sheet1.txt" FROM
"Spreadsheet1_Sheet1" WHERE Column2 = 'Bob'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Column1 INTO
"csv://Spreadsheet1_Sheet1.txt;delimiter=tab" FROM
"Spreadsheet1_Sheet1" WHERE Column2 = 'Bob'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )
```

```
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Spreadsheet1_Sheet1 (Column1) VALUES
(?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Spreadsheet1_Sheet1 SET Column1='John' WHERE
Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "6");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:googlesheets:InitiateOAuth=GETAN
DREFRESH;",);
String cmd = "DELETE FROM Spreadsheet1_Sheet1 WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "6");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
```

```
| <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Using Spreadsheets as Tables

Spreadsheets as Tables

The adapter models spreadsheets and ranges as relational tables.

Tables

Tables shows various configuration options to reflect your spreadsheets' organization in the tables; for example, you will find guides for working with headers and querying ranges as tables.

Columns provides more information on column discovery.

Stored Procedures

In addition to working with the data in the spreadsheet, you can use the available stored procedures to access functionality in the Google Sheets API that is not modeled as SELECT, INSERT, UPDATE, or DELETE statements.

Tables

The adapter enables you to represent a top-left-oriented spreadsheet or a user-specified range as a database table. You can control how tables are listed by setting the Header property.

Top-Left Oriented Tables

You can use the adapter to start working right away with top-left-oriented tables. The default configuration settings are explained below:

Top-left-oriented tables are represented with the name of the worksheet.

The default format requires that the table is top-left-oriented and that the first row of data in the worksheet contains the column names. This means that the default value of true for the Header connection string property is required.

Headers should not contain special characters.

By default the adapter will return all rows until the first empty row. *Note*: an empty row between data will prevent further data from being returned.

Due to a limitation of Google's Spreadsheet API, all column headers must be non empty.

User-Specified Range

You can execute SQL commands against a given range as a table by using this format in your query: *WORKSHEET#RANGE*

Note: Range notation is only available in a SELECT or UPDATE statement. Ranges are not supported for DELETE and INSERT commands.

Columns

You can specify column names or generate column names automatically by setting the Header property. This property affects how you use columns in commands.

Header=True (Default)

Columns are determined by the first row of the Google spreadsheet. If no values are provided for the first row of the spreadsheet, the adapter will create unique, alphabetized column names that are available only within the scope of that request.

The adapter also adds an Id column for each row that corresponds to the unique URI of the row on the Google servers. This is used during update and delete operations.

Header=False

Columns will be dynamically assigned based on either the specified range or the size of the worksheet. The autogenerated column names are alphabetical.

The Id column for each row will represent the row number from the top of the sheet. For example, if you specify a range A3:E6, rows 3, 4, 5, and 6 will be returned.

Views

Views are similar to tables in the way that data is represented; however, views do not support updates. Data sources that are represented as views are typically read-only data sources. Often, a stored procedure is available to update the data if such functionality is applicable to the data source.

Queries can be executed against a view as if it were a normal table, and the data that comes back is similar in that regard. To find out more about tables and stored procedures, please navigate to their corresponding entries in this help document.

Views are composed of columns and pseudo columns.

Google Sheets Adapter Views

Name	Description
Sheets	Returns a list of a user's sheets and their relevant information.
Spreadsheets	Returns a list of a user's spreadsheets and their relevant information.

Sheets

Returns a list of a user's sheets and their relevant information.

Columns

Name	Type	Description
SpreadsheetId	*String*	The assigned Id of the spreadsheet.
SpreadsheetName	*String*	The name of the spreadsheet.

SheetId	*String*	A short description of the spreadsheet.
SheetName	*String*	The name of the sheet.
SheetIndex	*Integer*	The index of the sheet within the spreadsheet.
SheetType	*String*	The type of sheet. Defaults to GRID.
RowCount	*Integer*	The number of rows in the grid.
ColumnCount	*Integer*	The number of columns in the grid.
FrozenRowCount	*Integer*	The number of rows that are frozen in the grid.
FrozenColumnCount	*Integer*	The number of columns that are frozen in the grid.

Spreadsheets

Returns a list of a user's spreadsheets and their relevant information.

Columns

Name	Type	Description
Id [KEY]	*String*	The assigned Id of the spreadsheet.
Name	*String*	The name of the spreadsheet.
Description	*String*	A short description of the spreadsheet.
Owners	*String*	The owners of the spreadsheet.
ModifiedTime	*Datetime*	The last updated date and time of this spreadsheet.
CreatedTime	*Datetime*	The created date and time of this spreadsheet.
Trashed	*Boolean*	Whether the spreadsheet has been trashed.
Starred	*Boolean*	Whether the user has starred the spreadsheet.
Hidden	*Boolean*	Whether the spreadsheet is hidden.

Viewed	*Boolean*	Whether the user has viewed the spreadsheet.

Stored Procedures

Stored Procedures are available to complement the data available from the Using Spreadsheets as Tables. Sometimes it is necessary to update data available from a view using a stored procedure because the data does not provide for direct, table-like, two-way updates. In these situations, the retrieval of the data is done using the appropriate view or table, while the update is done by calling a stored procedure. Stored procedures take a list of parameters and return back a dataset that contains the collection of tuples that constitute the response.

Google Sheets Adapter Stored Procedures

Name	Description
AddSheet	Add a worksheet to an existing Google spreadsheet.
Authenticate	Authenticate to the Google Spreadsheets service.
CreateSchema	Creates a schema file for the specified table or view.
CreateSpreadsheet	Creates a spreadsheet in the user's Google Drive.
DeleteSheet	Deletes a worksheet in an existing Google spreadsheet.
DeleteSpreadsheet	Deletes a spreadsheet.
DownloadDocument	Downloads a file from the user's Google Drive.
GetOAuthAccessToken	Obtains the OAuth access token to be used for authentication with various Google services.
GetOAuthAuthorizationURL	Obtains the OAuth authorization URL for authentication with various Google services.
RefreshOAuthAccessToken	Obtains the OAuth access token to be used for authentication with various Google services.

UpdateSheet	Add a worksheet to an existing Google spreadsheet.
UploadDocument	Uploads a file to the user's Google Drive.

AddSheet

Add a worksheet to an existing Google spreadsheet.

Input

Name	Type	Description
SpreadsheetId	*String*	The ID of the spreadsheet.
SheetId	*String*	The ID of the sheet. Must be non-negative. This field cannot be changed once set.
Title	*String*	The name of the sheet.
Index	*String*	The index of the sheet within the spreadsheet.
SheetType	*String*	The type of sheet. Defaults to GRID. This field cannot be changed once set. The allowed values are *GRID, OBJECT*. The default value is *GRID*.
RowCount	*String*	The number of rows in the grid.
ColumnCount	*String*	The number of columns in the grid.
FrozenRowCount	*String*	The number of rows that are frozen in the grid.
FrozenColumnCount	*String*	The number of columns that are frozen in the grid.
HideGridlines	*String*	True if the grid is not showing gridlines in the UI. The allowed values are *true, false*.

Hidden	*String*	True if the sheet is hidden in the UI, false if it is visible. The allowed values are *true, false*.
RightToLeft	*String*	True if the sheet is an RTL sheet instead of an LTR sheet. The allowed values are *true, false*.

Output

Name	Type	Description
Success	*String*	This value shows whether the operation was successful or not.

Authenticate

Authenticate to the Google Spreadsheets service.

Input

Name	Type	Description
User	*String*	The username used to authenticate with Google.
Password	*String*	The password used to authenticate with Google.
Service	*String*	The service identifier to authenticate with (e.g., Blogger, Calendar, Docs, or Spreadsheets).

Output

Name	Type	Description
AuthToken	*String*	The authentication token returned from Google. This can be used in subsequent calls to other operations for this particular service.

CreateSchema

Creates a schema file for the specified table or view.

Input

Name	Type	Description
TableName	*String*	The name of the table or view.
FileName	*String*	The full file path and name of the schema to generate.

Output

Name	Type	Description
Result	*String*	Returns Success or Failure.

CreateSpreadsheet

Creates a spreadsheet in the user's Google Drive.

Input

Name	Type	Description
Title	*String*	The title for the spreadsheet.
Description	*String*	The description for the spreadsheet.
Hidden	*String*	This parameter sets whether or not the resource is hidden. The allowed values are *TRUE*, *FALSE*. The default value is *FALSE*.

Restricted	*String*	This parameter sets whether or not the resource is restricted. The allowed values are *TRUE, FALSE*. The default value is *FALSE*.
Starred	*String*	This parameter sets whether or not the resource is starred. The allowed values are *TRUE, FALSE*. The default value is *FALSE*.
Parents	*String*	The Ids of the parent folders for the created spreadsheet.

Output

Name	Type	Description
Success	*String*	This parameter indicates whether the operation was successful or not.
Id	*String*	The Id of the new spreadsheet.

DeleteSheet

Deletes a worksheet in an existing Google spreadsheet.

Input

Name	Type	Description
SpreadsheetId	*String*	The ID of the spreadsheet.
SheetId	*String*	The ID of the sheet.

Output

Name	Type	Description
Success	*String*	This value shows whether the operation was successful or not.

DeleteSpreadsheet

Deletes a spreadsheet.

Input

Name	Type	Description
SpreadsheetId	*String*	The ID of the spreadsheet.

Output

Name	Type	Description
Success	*String*	This value shows whether the operation was successful or not.

DownloadDocument

Downloads a file from the user's Google Drive.

Input

Name	Type	Description
Id	*String*	The Id of the resource to be downloaded.

FileFormat	*String*	The file format to be applied when saving the file, such as text/plain. The default value is *application/vnd.openxmlformats-officedocument.spreadsheetml.sheet*.
LocalFile	*String*	The local file path including the file name for the location where the file will be saved on disk. Leave empty to keep the file in memory.
Encoding	*String*	If the LocalFile input is left empty, the data will be output to FileData in the specified encoding. The allowed values are *NONE*, *BASE64*. The default value is *BASE64*.
Overwrite	*String*	What to do when downloaded file exists. Set true to overwrite. The allowed values are *true*, *false*. The default value is *false*.

Output

Name	Type	Description
FileData	*String*	If the LocalFile input is empty, file data will be output in the format specified by the Encoding input.
Success	*String*	This value shows a boolean indication of whether the operation was successful or not.

GetOAuthAccessToken

Obtains the OAuth access token to be used for authentication with various Google services.

Input

Name	Type	Description
AuthMode	*String*	The type of authentication mode to use. The allowed values are *APP, WEB*. The default value is *WEB*.
Verifier	*String*	The verifier code returned by Google after permissions have been granted for the app to connect. WEB AuthMode only.
Scope	*String*	The scope of access to Google APIs. By default, access to all APIs used by this data provider will be specified. The default value is *https://www.googleapis.com/auth/drive https://www.googleapis.com/auth/spreadsheets*.
CallbackURL	*String*	Determines where the response is sent. The value of this parameter must exactly match one of the values registered in the APIs Console (including the HTTP or HTTPS schemes, capitalization, and trailing '/').
ApprovalPrompt	*String*	Indicates if the user should be re-prompted for consent. The default is AUTO, so a given user should only see the consent page for a given set of scopes the first time through the sequence. If the value is FORCE, then the user sees a consent page even if they have previously given consent to your application for a given set of scopes. The allowed values are *AUTO, FORCE*. The default value is *AUTO*.

AccessType	*String*	Indicates if your application needs to access a Google API when the user is not present at the browser. This parameter defaults to OFFLINE. If your application needs to refresh access tokens when the user is not present at the browser, then use OFFLINE. This will result in your application obtaining a refresh token the first time your application exchanges an authorization code for a user. The allowed values are *ONLINE*, *OFFLINE*. The default value is *OFFLINE*.
State	*String*	Indicates any state which may be useful to your application upon receipt of the response. Your application receives the same value it sent, as this parameter makes a round-trip to the Google authorization server and back. Uses include redirecting the user to the correct resource in your site, using nonces, and mitigating cross-site request forgery.

Output

Name	Type	Description
OAuthAccessToken	*String*	The authentication token returned from Google. This can be used in subsequent calls to other operations for this particular service.
OAuthRefreshToken	*String*	A token that may be used to obtain a new access token.
ExpiresIn	*String*	The remaining lifetime on the access token.

GetOAuthAuthorizationURL

Obtains the OAuth authorization URL for authentication with various Google services.

Input

Name	Type	Description
Scope	*String*	The scope of access to Google APIs. By default, access to all APIs used by this data provider will be specified. The default value is *https://www.googleapis.com/auth/drive https://www.googleapis.com/auth/spreadsheets*.
CallbackURL	*String*	Determines where the response is sent. The value of this parameter must exactly match one of the values registered in the APIs Console (including the HTTP or HTTPS schemes, capitalization, and trailing '/').
ApprovalPrompt	*String*	Indicates if the user should be re-prompted for consent. The default is AUTO, so a given user should only see the consent page for a given set of scopes the first time through the sequence. If the value is FORCE, then the user sees a consent page even if they have previously given consent to your application for a given set of scopes. The allowed values are *AUTO, FORCE*. The default value is *AUTO*.
AccessType	*String*	Indicates if your application needs to access a Google API when the user is not present at the browser. This parameter defaults to OFFLINE. If your application needs to refresh access tokens when the user is not present at the browser, then use OFFLINE. This will result in your application obtaining a refresh token the first time your application exchanges an authorization code for a user. The allowed values are *ONLINE, OFFLINE*. The default value is *OFFLINE*.

State	*String*	Indicates any state which may be useful to your application upon receipt of the response. Your application receives the same value it sent, as this parameter makes a round-trip to the Google authorization server and back. Uses include redirecting the user to the correct resource in your site, nonces, and cross-site-request-forgery mitigations.

Output

Name	Type	Description
URL	*String*	The URL to complete user authentication.

RefreshOAuthAccessToken

Obtains the OAuth access token to be used for authentication with various Google services.

Input

Name	Type	Description
OAuthRefreshToken	*String*	The refresh token returned from the original authorization code exchange.

Output

Name	Type	Description

OAuthAccessToken	*String*	The authentication token returned from Google. This can be used in subsequent calls to other operations for this particular service.
ExpiresIn	*String*	The remaining lifetime on the access token.

UpdateSheet

Add a worksheet to an existing Google spreadsheet.

Input

Name	Type	Description
SpreadsheetId	*String*	The ID of the spreadsheet.
SheetId	*String*	The ID of the sheet. Must be non-negative.
Title	*String*	The name of the sheet.
Index	*String*	The index of the sheet within the spreadsheet.
RowCount	*String*	The number of rows in the grid.
ColumnCount	*String*	The number of columns in the grid.
FrozenRowCount	*String*	The number of rows that are frozen in the grid.
FrozenColumnCount	*String*	The number of columns that are frozen in the grid.
HideGridlines	*String*	True if the grid is not showing gridlines in the UI. The allowed values are *true, false*.

Hidden	*String*	True if the sheet is hidden in the UI, false if it is visible. The allowed values are *true, false*.
RightToLeft	*String*	True if the sheet is an RTL sheet instead of an LTR sheet. The allowed values are *true, false*.

Output

Name	Type	Description
Success	*String*	This value shows whether the operation was successful or not.

UploadDocument

Uploads a file to the user's Google Drive.

Input

Name	Type	Description
Id	*String*	The Id for the file. Only needs to be set when updating an existing document.
Name	*String*	The name for the file, including the extension.
Description	*String*	The description for the file.
Starred	*String*	This parameter sets whether or not the resource is starred. The allowed values are *TRUE, FALSE*. The default value is *FALSE*.

Parents	*String*	The Ids of the parent folders for the uploaded document.
MIMEType	*String*	The MIME type of the file. The default value is *application/vnd.google-apps.spreadsheet*.
LocalFile	*String*	The local file path including the file name of the file to be uploaded. A value for this field is required when FileData is not specified.
FileData	*String*	If the LocalFile input is empty, the file data will be output to a file in the format specified by the Encoding parameter.
Encoding	*String*	The FileData input encoding type. The allowed values are *NONE*, *BASE64*. The default value is *BASE64*.

Output

Name	Type	Description
Id	*String*	The id of the file.
Success	*String*	This parameter sets whether the operation was successful or not.

System Tables

Query the following system tables to access schema information, information on data source functionality, and batch operation statistics.

Schema Tables

The following tables return database metadata for Google Sheets:

sys_catalogs: Lists the available databases.

sys_schemas: Lists the available schemas.

sys_tables: Lists the available tables.

sys_tablecolumns: Describes the columns of the available tables.

sys_views: Lists the available views.

sys_viewcolumns: Describes the columns of available views.

sys_procedures: Describes the available stored procedures.

sys_procedureparameters: Describes stored procedure parameters.

sys_keycolumns: Describes the primary and foreign keys.

sys_indexes: Describes the available indexes.

Data Source Tables

The following tables return information about how to connect to and query the data source:

sys_connection_props: Returns information on the available connection properties.

sys_sqlinfo: Describes the SELECT queries that the adapter can offload to the data source.

Query Information Tables

Query statistics are available for data modification queries

sys_identity: Returns information about batch operations or single updates.

sys_catalogs

Lists the available databases.

The following query retrieves all databases determined by the connection string:

```
SELECT * FROM sys_catalogs
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.

sys_schemas

Lists the available schemas.

```
SELECT * FROM sys_schemas
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.
SchemaName	*String*	The schema name.

sys_tables

Lists the available tables.

```
SELECT * FROM sys_tables
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table.
TableType	*String*	The table type.
Description	*String*	A description of the table.

sys_tablecolumns

Describes the columns of the available tables.

The following query returns the columns and data types for the Spreadsheet1_Sheet1 table:

```
SELECT ColumnName, DataTypeName FROM sys_tablecolumns WHERE
TableName='Spreadsheet1_Sheet1'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table containing the column.
ColumnName	*String*	The column name.
DataTypeName	*String*	The data type name.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	A brief description of the column.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.

IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_views

Lists the available views.

```
SELECT TableName FROM sys_views
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
TableType	*String*	The type of the view.
Description	*String*	A description of the view.

sys_viewcolumns

Describes the columns of the available views.

The following query returns the columns and data types for a specified view:

```
SELECT ColumnName, DataTypeName FROM sys_viewolumns WHERE
TableName='MyView'
```

Columns

Name	Type	Description

CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
ColumnName	*String*	The name of the column.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type of the column. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	The column description.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_procedures

Lists the available stored procedures.

```
SELECT * FROM sys_procedures
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the stored procedure.
SchemaName	*String*	The schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure.
Description	*String*	A description of the stored procedure.

sys_procedureparameters

Describes stored procedure parameters.

The following query returns information about all of the input parameters for the DownloadDocument stored procedure:

```
SELECT * FROM sys_procedureparameters WHERE
ProcedureName='DownloadDocument' AND Direction=1 OR Direction=2
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the stored procedure.
SchemaName	*String*	The name of the schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure containing the parameter.
ColumnName	*String*	The name of the stored procedure parameter.
Direction	*Int32*	An integer corresponding to the type of the parameter: input (1), input/output (2), or output(4). input/output type parameters can be both input and output parameters.

DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The number of characters allowed for character data. The number of digits allowed for numeric data.
NumericPrecision	*Int32*	The maximum precision for numeric data. The column length in characters for character and date-time data.
NumbericScale	*Int32*	The number of digits to the right of the decimal point in numeric data.
IsNullable	*Boolean*	Whether the parameter can contain null.
Description	*String*	The description of the parameter.
Ordinal	*Int32*	The index of the parameter.

sys_keycolumns

Describes the primary and foreign keys.

The following query retrieves the primary key for the Spreadsheet1_Sheet1 table:

```
SELECT * FROM sys_keycolumns WHERE IsKey='True' AND
TableName='Spreadsheet1_Sheet1'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the key.
SchemaName	*String*	The name of the schema containing the key.
TableName	*String*	The name of the table containing the key.
ColumnName	*String*	The name of the key column.

IsKey	*Boolean*	Whether the column is a primary key in the table referenced in the TableName table.
IsForeignKey	*Boolean*	Whether the column is a foreign key referenced in the TableName table.
ReferencedCatalogName	*String*	The database containing the primary key.
ReferencedSchemaName	*String*	The schema containing the primary key.
ReferencedTableName	*String*	The table containing the primary key.
ReferencedColumnName	*String*	The column name of the primary key.

sys_indexes

Describes the available indexes. By filtering on indexes, you can write more selective queries with faster query response times.

The following query retrieves all indexes that are not primary keys:

```
SELECT * FROM sys_indexes WHERE IsPrimary='false'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the index.
SchemaName	*String*	The name of the schema containing the index.
TableName	*String*	The name of the table containing the index.
IndexName	*String*	The index name.
ColumnName	*String*	The name of the column associated with the index.

IsUnique	*Boolean*	True if the index is unique. False otherwise.
IsPrimary	*Boolean*	True if the index is a primary key. False otherwise.
Type	*Int16*	An integer value corresponding to the index type: statistic (0), clustered (1), hashed (2), or other (3).
SortOrder	*String*	The sort order: A for ascending or D for descending.
OrdinalPosition	*Int16*	The sequence number of the column in the index.

sys_connection_props

Returns information on the available connection properties and those set in the connection string.

The following query retrieves all connection properties that have been set in the connection string or set through a default value:

```
SELECT * FROM sys_connection_props WHERE Value <> ''
```

Columns

Name	Type	Description
Name	*String*	The name of the connection property.
ShortDescription	*String*	A brief description.
Type	*String*	The data type of the connection property.
Default	*String*	The default value if one is not explicitly set.
Values	*String*	A comma-separated list of possible values. A validation error is thrown if another value is specified.
Value	*String*	The value you set or a preconfigured default.
Required	*Boolean*	Whether the property is required to connect.

Category	*String*	The category of the connection property.
IsSessionProperty	*String*	Whether the property is a session property, used to save information about the current connection.

sys_sqlinfo

Describes the SELECT query processing that the adapter can offload to the data source.

Collaborative Query Processing

When working with data sources that do not support SQL-92, you can query the sys_sqlinfo view to determine the query capabilities of the underlying APIs, expressed in SQL syntax. See SQL Compliance for SQL syntax details.

Discovering the Data Source's SELECT Capabilities

Below is an example data set of SQL capabilities. Some aspects of SELECT functionality are returned in a comma-separated list if supported; otherwise, the column contains NO.

Name	**Possible Values**
AGGREGATE_FUNCTIONS	AVG, COUNT, MAX, MIN, SUM, DISTINCT
COUNT	YES
SUPPORTED_OPERATORS	=, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR
GROUP_BY	NO_RELATION
OUTER_JOINS	YES
OJ_CAPABILITIES	NESTED, LEFT, RIGHT, INNER, NOT_ORDERED, ALL_COMPARISON_OPS

SUBQUERIES	COMPARISON, EXISTS, IN, CORRELATED_SUBQUERIES, QUANTIFIED
STRING_FUNCTIONS	ASCII,CHAR,CONCAT,LEFT,LTRIM,REPLACE,RIGHT,RTRIM,SOUNDEX,SPACE,SUBSTRING
NUMERIC_FUNCTIONS	ABS,ACOS,ASIN,ATAN,CEILING,COS,COT,DEGREES,EXP,FLOOR,LOG,LOG10,PI,POWER,RADIANS,RAND,ROUND,SIGN,SIN,SQRT,TAN
TIMEDATE_FUNCTIONS	CURRENT_DATE,CURRENT_TIMESTAMP,MONTH,YEAR
IDENTIFIER_QUOTE_OPEN_CHAR	[
IDENTIFIER_QUOTE_CLOSE_CHAR	]

The following query retrieves the operators that can be used in the WHERE clause:

```
SELECT * FROM sys_sqlinfo WHERE Name='SUPPORTED_OPERATORS'
```

Note that individual tables may have different limitations or requirements on the WHERE clause; refer to the Using Spreadsheets as Tables section for more information.

Columns

Name	Type	Description
NAME	*String*	A component of SQL syntax, or a capability that can be processed on the server.
VALUE	*String*	Detail on the supported SQL or SQL syntax.

sys_identity

Returns information about attempted modifications. The following query retrieves the Ids of the modified rows in a batch operation.

```
SELECT * FROM sys_identity
```

Columns

Name	Type	Description
Id	*String*	The database-generated Id returned from a data modification operation.
Batch	*String*	An identifier for the batch. 1 for a single operation.
Operation	*String*	The result of the operation in the batch: INSERTED, UPDATED, or DELETED.
Message	*String*	SUCCESS or an error message if the update in the batch failed.

TDV HubSpot Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

HubSpot uses the OAuth authentication standard. See Connecting to HubSpot for a guide to completing the process.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Column Sizes	A file with a name=value comma separated list of column sizes. For use with custom fields.
Company Properties File	A file with a comma separated list of properties to select from HubSpot for the Companies table.
Contact Properties File	A file with a comma separated list of properties to select from HubSpot for the Contacts table.
Deal Properties File	A file with a comma separated list of properties to select from HubSpot for the Deals table.

Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to HubSpot from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Display Names	Boolean determining if the display names for the columns should be used instead of the API names.
Use Simple Names	Boolean determining if simple names should be used for tables and columns.

Callback URL

The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.

Data Type

string

Default Value

""

Remarks

During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

Column Sizes

A file with a name=value comma separated list of column sizes. For use with custom fields.

Data Type

string

Default Value

""

Remarks

Some custom fields may requre a larger column size than the default. These can be entered as name=value pairs. For instance: Col1=4000,Col2=5000,Col3=13. The values entered will override the default for any detected custom fields.

Company Properties File

A file with a comma separated list of properties to select from HubSpot for the Companies table.

Data Type

string

Default Value

""

Remarks

In HubSpot, you can potentially have so many custom fields that attempting to select all of them at a time will cause HubSpot to throw an error. In a situation where you do not have direct control over the SQL Statement but still need to get custom field data back, you set the CompanyPropertiesFile to the full location of a file that lists which custom fields to select. For instance, C:\users\public\documents\customfields.txt.

The CustomFields document itself is just a comma separated list of the names of which custom fields to select. Note that these must be the API names of the custom fields. These names can be retrieved from the Name column of the DealProperties table. For instance: dealname,amount,description.

Contact Properties File

A file with a comma separated list of properties to select from HubSpot for the Contacts table.

Data Type

string

Default Value

""

Remarks

In HubSpot, you can potentially have so many custom fields that attempting to select all of them at a time will cause HubSpot to throw an error. In a situation where you do not have direct control over the SQL Statement but still need to get custom field data back, you set the ContactPropertiesFile to the full location of a file that lists which custom fields to select. For instance, C:\users\public\documents\customfields.txt.

The CustomFields document itself is just a comma separated list of the names of which custom fields to select. Note that these must be the API names of the custom fields. These names can be retrieved from the Name column of the ContactProperties table. For instance: firstname,lastname,lastmodifieddate.

Deal Properties File

A file with a comma separated list of properties to select from HubSpot for the Deals table.

Data Type

string

Default Value

""

Remarks

In HubSpot, you can potentially have so many custom fields that attempting to select all of them at a time will cause HubSpot to throw an error. In a situation where you do not have direct control over the SQL Statement but still need to get custom field data back, you set the DealPropertiesFile to the full location of a file that lists which custom fields to select. For instance, C:\users\public\documents\customfields.txt.

The CustomFields document itself is just a comma separated list of the names of which custom fields to select. Note that these must be the API names of the custom fields. These names can be retrieved from the Name column of the DealProperties table. For instance: dealname,amount,description.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to HubSpot and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\HubSpot Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
```

```
domain\user
```

Readonly

You can use this property to enforce read-only access to HubSpot from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example

A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801c bb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Display Names

Boolean determining if the display names for the columns should be used instead of the API names.

Data Type

bool

Default Value

true

Remarks

Boolean determining if the display names for the columns should be used instead of the API names.

Use Simple Names

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

Boolean determining if simple names should be used for tables and columns. This will only affect custom fields, which may have non-standard SQL characters in them. UseSimpleNames makes the adapter easier to use with traditional database tools.

Setting UseSimpleNames to true will simplify the names of custom field columns returned. If set to false, the custom field columns will appear as they do in HubSpot.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the HubSpot Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to HubSpot

OAuth requires the authenticating user to interact with HubSpot using the browser. The adapter facilitates this in various ways as described below.

Authenticate to HubSpot

After setting InitiateOAuth to GETANDREFRESH, you are ready to connect. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the OAuthAccessToken connection property. When you connect the adapter opens the OAuth endpoint in your default browser. Log in and grant permissions to the application. The adapter then completes the OAuth process:

Extracts the access token from the callback URL and authenticates requests.

Refreshes the access token when it expires.

Saves OAuth values in

OAuthSettingsLocation to be persisted across connections.

Advanced Settings

Accessing Custom Fields

If you have defined a large number of custom fields, *SELECT* * queries can fail with an error as HubSpot limits how many fields can be requested. To avoid the error, you can specify the custom fields you want to return in the text files specified by ContactPropertiesFile and DealPropertiesFile. Alternatively, specify the primary key in the WHERE clause.

If the detected column sizes for custom fields are not sufficient, you can specify the sizes in ColumnSizes.

Fine-Tuning Data Access

You can use the following properties to gain more control over column names:

UseDisplayNames

UseSimpleNames

Create an App

The HubSpot Adapter is already registered with HubSpot; to display your own information to users when they log in, follow the steps below to register an app: Follow the steps below to obtain the OAuth client credentials, the OAuthClientId and OAuthClientSecret:

Log into your HubSpot developer account.

Click Create application.

If you will only use the app to connect to your portal, select Private.

If other users will use the app to connect to their own portals, select Public.

Click the name of your app (or click edit).

Enter values to be displayed to users when you connect. These values include the app name, author name, and a description of the app.

Make note of or copy the Client ID and Client Secret to be used for the

OAuthClientId and OAuthClientSecret properties.

The adapter enables the granular control useful in more complex integrations or network topologies; for example, connecting from behind a firewall. In this section you can find more advanced adapter configurations, as well as steps to troubleshoot connection errors.

Connecting Through a Firewall or Proxy

HTTP Proxies

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties: Set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, additionally set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The HubSpot Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the HubSpot API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the HubSpot adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
```

```
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Contacts
```

Rename a column:

```
SELECT "Country" AS MY_Country FROM Contacts
```

Search data:

```
SELECT * FROM Contacts WHERE VID = '123456789';
```

The HubSpot APIs support the following operators in the WHERE clause: =, AND, OR.

```
SELECT * FROM Contacts WHERE VID = '123456789';
```

Sort a result set in ascending order:

```
SELECT City, Country FROM Contacts  ORDER BY Country ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT City, Country INTO "csv://Contacts.txt" FROM "Contacts"
WHERE VID = '123456789'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT City, Country INTO "csv://Contacts.txt;delimiter=tab" FROM
"Contacts" WHERE VID = '123456789'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Contacts (Country) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "Canada");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Contacts SET Country='Canada' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "123456789");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:hubspot:InitiateOAuth=GETANDREFR
ESH;",);
String cmd = "DELETE FROM Contacts WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "123456789");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV JDBC-ODBC Bridge Adapter

Requirements and Restrictions

The restrictions are dependent on the ODBC driver being connected to.

Basic Tab

To connect to an ODBC data source, specify either the DSN (data source name) or specify an ODBC connection string: Set Driver and the connection properties for your ODBC driver.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Driver	The ODBC Driver to connect to.
DSN	The ODBC DSN to connect to.
Password	The database account password.
User	The database user account id.

Driver

The ODBC Driver to connect to.

Data Type

string

Default Value

""

Remarks

This property is used to specify the ODBC Driver to use, along with any connection properties that are required.

For example: {SQL Server};Server=myServerAddress;Database=myDataBase;Uid=myUsername;Pwd=myPassword;

DSN

The ODBC DSN to connect to.

Data Type

string

Default Value

""

Remarks

This property is used to specify the name of the ODBC DSN to use.

Set this property to the name as it is listed within the ODBC Data Source Administrator on Windows or the odbc.ini file on other operating systems.

Password

The database account password.

Data Type

string

Default Value

""

Remarks

This field maps to PWD connection property.

User

The database user account id.

Data Type

string

Default Value

""

Remarks

This field maps to UID connection property.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the JDBC-ODBC Bridge Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

System Tables

Query the following system tables to access schema information, information on data source functionality, and batch operation statistics.

Schema Tables

The following tables return database metadata for JDBC-ODBC Bridge:

sys_catalogs

: Lists the available databases.

sys_schemas

: Lists the available schemas.

sys_tables

: Lists the available tables.

sys_tablecolumns

: Describes the columns of the available tables.

sys_views

: Lists the available views.

sys_viewcolumns

: Describes the columns of available views.

sys_procedures

: Describes the available stored procedures.

sys_procedureparameters

: Describes stored procedure parameters.

sys_keycolumns

: Describes the primary and foreign keys.

sys_indexes

: Describes the available indexes.

Data Source Tables

The following tables return information about how to connect to and query the data source:

sys_connection_props

: Returns information on the available connection properties.

sys_sqlinfo

: Describes the SELECT queries that the adapter can offload to the data source.

Query Information Tables

Query statistics are available for data modification queries, including batch operations.

sys_identity

: Returns information about batch operations or single updates.

sys_catalogs

Lists the available databases.

The following query retrieves all databases determined by the connection string:

```
SELECT * FROM sys_catalogs
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.

sys_schemas

Lists the available schemas.

```
SELECT * FROM sys_schemas
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.
SchemaName	*String*	The schema name.

sys_tables

Lists the available tables.

```
SELECT * FROM sys_tables
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table.
TableType	*String*	The table type.
Description	*String*	A description of the table.

sys_tablecolumns

Describes the columns of the available tables.

The following query returns the columns and data types for the Account table:

```
SELECT ColumnName, DataTypeName FROM sys_tablecolumns WHERE
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table containing the column.
ColumnName	*String*	The column name.
DataTypeName	*String*	The data type name.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.

NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	A brief description of the column.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_views

Lists the available views.

```
SELECT TableName FROM sys_views
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.

TableType	*String*	The type of the view.
Description	*String*	A description of the view.

sys_viewcolumns

Describes the columns of the available views.

The following query returns the columns and data types for a specified view:

```
SELECT ColumnName, DataTypeName FROM sys_viewolumns WHERE
TableName='MyView'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
ColumnName	*String*	The name of the column.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type of the column. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	The column description.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.

IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_procedures

Lists the available stored procedures.

```
SELECT * FROM sys_procedures
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the stored procedure.
SchemaName	*String*	The schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure.
Description	*String*	A description of the stored procedure.

sys_procedureparameters

Describes stored procedure parameters.

The following query returns information about all of the input parameters for the SelectEntries stored procedure:

```
SELECT * FROM sys_procedureparameters WHERE
ProcedureName='SelectEntries' AND Direction=1 OR Direction=2
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the stored procedure.
SchemaName	*String*	The name of the schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure containing the parameter.
ColumnName	*String*	The name of the stored procedure parameter.
Direction	*Int32*	An integer corresponding to the type of the parameter: input (1), input/output (2), or output(4). input/output type parameters can be both input and output parameters.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The number of characters allowed for character data. The number of digits allowed for numeric data.
NumericPrecision	*Int32*	The maximum precision for numeric data. The column length in characters for character and date-time data.
NumbericScale	*Int32*	The number of digits to the right of the decimal point in numeric data.
IsNullable	*Boolean*	Whether the parameter can contain null.
Description	*String*	The description of the parameter.
Ordinal	*Int32*	The index of the parameter.

sys_keycolumns

Describes the primary and foreign keys.

The following query retrieves the primary key for the Account table:

```
SELECT * FROM sys_keycolumns WHERE IsKey='True' AND
TableName='Account'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the key.
SchemaName	*String*	The name of the schema containing the key.
TableName	*String*	The name of the table containing the key.
ColumnName	*String*	The name of the key column.
IsKey	*Boolean*	Whether the column is a primary key in the table referenced in the TableName table.
IsForeignKey	*Boolean*	Whether the column is a foreign key referenced in the TableName table.
ReferencedCatalogName	*String*	The database containing the primary key.
ReferencedSchemaName	*String*	The schema containing the primary key.
ReferencedTableName	*String*	The table containing the primary key.
ReferencedColumnName	*String*	The column name of the primary key.

sys_indexes

Describes the available indexes. By filtering on indexes, you can write more selective queries with faster query response times.

The following query retrieves all indexes that are not primary keys:

```
SELECT * FROM sys_indexes WHERE IsPrimary='false'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the index.
SchemaName	*String*	The name of the schema containing the index.

TableName	*String*	The name of the table containing the index.
IndexName	*String*	The index name.
ColumnName	*String*	The name of the column associated with the index.
IsUnique	*Boolean*	True if the index is unique. False otherwise.
IsPrimary	*Boolean*	True if the index is a primary key. False otherwise.
Type	*Int16*	An integer value corresponding to the index type: statistic (0), clustered (1), hashed (2), or other (3).
SortOrder	*String*	The sort order: A for ascending or D for descending.
OrdinalPosition	*Int16*	The sequence number of the column in the index.

sys_connection_props

Returns information on the available connection properties and those set in the connection string.

The following query retrieves all connection properties that have been set in the connection string or set through a default value:

```
SELECT * FROM sys_connection_props WHERE Value <> ''
```

Columns

Name	Type	Description
Name	*String*	The name of the connection property.
ShortDescription	*String*	A brief description.
Type	*String*	The data type of the connection property.
Default	*String*	The default value if one is not explicitly set.

Values	*String*	A comma-separated list of possible values. A validation error is thrown if another value is specified.
Value	*String*	The value you set or a preconfigured default.
Required	*Boolean*	Whether the property is required to connect.
Category	*String*	The category of the connection property.
IsSessionProperty	*String*	Whether the property is a session property, used to save information about the current connection.

sys_sqlinfo

Describes the SELECT query processing that the adapter can offload to the data source.

Collaborative Query Processing

When working with data sources that do not support SQL-92, you can query the sys_sqlinfo view to determine the query capabilities of the underlying APIs, expressed in SQL syntax.

Discovering the Data Source's SELECT Capabilities

Below is an example data set of SQL capabilities. Some aspects of SELECT functionality are returned in a comma-separated list if supported; otherwise, the column contains NO.

Name	**Possible Values**
AGGREGATE_FUNCTIONS	AVG, COUNT, MAX, MIN, SUM, DISTINCT
COUNT	YES
SUPPORTED_OPERATORS	=, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR
GROUP_BY	NO_RELATION

OUTER_JOINS	YES
OJ_CAPABILITIES	NESTED, LEFT, RIGHT, INNER, NOT_ORDERED, ALL_COMPARISON_OPS
SUBQUERIES	COMPARISON, EXISTS, IN, CORRELATED_SUBQUERIES, QUANTIFIED
STRING_FUNCTIONS	ASCII,CHAR,CONCAT,LEFT,LTRIM,REPLACE,RIGHT,RTRIM,SOUNDEX,SPACE,SUBSTRING
NUMERIC_FUNCTIONS	ABS,ACOS,ASIN,ATAN,CEILING,COS,COT,DEGREES,EXP,FLOOR,LOG,LOG10,PI,POWER,RADIANS,RAND,ROUND,SIGN,SIN,SQRT,TAN
TIMEDATE_FUNCTIONS	CURRENT_DATE,CURRENT_TIMESTAMP,MONTH,YEAR
IDENTIFIER_QUOTE_OPEN_CHAR	[
IDENTIFIER_QUOTE_CLOSE_CHAR	]

The following query retrieves the operators that can be used in the WHERE clause:

```
SELECT * FROM sys_sqlinfo WHERE Name='SUPPORTED_OPERATORS'
```

Note that individual tables may have different limitations or requirements on the WHERE clause; refer to the Data Model section for more information.

Columns

Name	Type	Description
NAME	*String*	A component of SQL syntax, or a capability that can be processed on the server.
VALUE	*String*	Detail on the supported SQL or SQL syntax.

sys_identity

Returns information about attempted modifications. The following query retrieves the Ids of the modified rows in a batch operation.

```
SELECT * FROM sys_identity
```

Columns

Name	Type	Description
Id	*String*	The database-generated Id returned from a data modification operation.
Batch	*String*	An identifier for the batch. 1 for a single operation.
Operation	*String*	The result of the operation in the batch: INSERTED, UPDATED, or DELETED.
Message	*String*	SUCCESS or an error message if the update in the batch failed.

TDV Marketo Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Both the REST and SOAP APIs are supported and can be chosen by using the Schema property.

For the REST API: The OAuthClientId, OAuthClientSecret, and RESTEndpoint properties, under the OAuth and REST Connection sections, must be set to valid Marketo user credentials. For more information, refer to our Connecting to the REST API guide.

For the SOAP API: The UserId, EncryptionKey, and SOAPEndpoint properties, under the SOAP Connection section, must be set to valid Marketo user credentials. For more information, refer to our Connecting to the SOAP API guide.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

The following is the full list of the options you can configure in the connection string for this provider.

Check Prompt Mode	For ODBC, check whether the application says if it supports prompting the user. You can turn this off to force it to launch the browser during OAuth. (Qlikview, Visual Studio, Power BI all have NO_PROMPT).
Encryption Key	The Marketo SOAP API Encryption Key.
Firewall Password	A password used to authenticate to a proxy-based firewall.

Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Job Polling Interval	Specifies the polling interval (in seconds) when checking the status of a bulk API job.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per page from Marketo.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Marketo from the provider.
REST Endpoint	The Marketo REST API Endpoint.
Schema	The type of schema to use.
SOAP Endpoint	The Marketo SOAP API Endpoint.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Bulk API	Specifies whether to use the Marketo Bulk API.
User Id	The Marketo SOAP API User Id.

Check Prompt Mode

For ODBC, check whether the application says if it supports prompting the user. You can turn this off to force it to launch the browser during OAuth. (Qlikview, Visual Studio, Power BI all have NO_PROMPT).

Data Type

bool

Default Value

false

Remarks

For ODBC, check whether the application says if it supports prompting the user. You can turn this off to force it to launch the browser during OAuth. (Qlikview, Visual Studio, Power BI all have NO_PROMPT).

For ODBC, check whether the application says if it supports prompting the user. You can turn this off to force it to launch the browser during OAuth. (Qlikview, Visual Studio, Power BI all have NO_PROMPT).

Encryption Key

The Marketo SOAP API Encryption Key.

Data Type

string

Default Value

""

Remarks

The EncryptionKey is generated on the Admin page of the Marketo website and is used to authenticate to the Marketo SOAP Web service.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Marketo and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"GETANDREFRESH"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Job Polling Interval

Specifies the polling interval (in seconds) when checking the status of a bulk API job.

Data Type

int

Default Value

120

Remarks

This property is used to specify the polling interval (in seconds) to identify when a bulk API job has completed. The adapter will wait JobPollingInterval seconds between calls to check a bulk API job status. Once the job is identified as 'Completed', the adapter will download and parse the generated file returning the results of the specified query.

This property can be set to 0 to just create and enqueue a job in which case the Job Id will be returned in the result set. The job status can then be checked using stored procedures.

Note: This property is only applicable when UseBulkAPI is set to True. See the UseBulkAPI page for more information about using the Bulk API.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\Marketo Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per page from Marketo.

Data Type

string

Default Value

"1000"

Remarks

The PageSize can control the number of results requested from Marketo on a given query. Setting a higher PageSize will cause more data to come back in a given request, but may take longer to execute. Setting a smaller PageSize is generally recommended to ensure timeout exceptions do not occur.

The default value is 1000; however, Marketo may impose a smaller items-per-page limit (such as 100). Therefore the PageSize will be 100 for all subsequent requests.

Note the Marketo REST API imposes a 300 items-per-page limit and thus 300 is the maximum number of items that will be returned in a Marketo REST API.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.

TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Marketo from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

REST Endpoint

The Marketo REST API Endpoint.

Data Type

string

Default Value

""

Remarks

The URL of the REST Web service endpoint is provided by Marketo on the Admin page of the Marketo website.

Schema

The type of schema to use.

Data Type

string

Default Value

"REST"

Remarks

The schemas available are REST (to use Marketo's REST API) and SOAP (to use Marketo's SOAP API).

SOAP Endpoint

The Marketo SOAP API Endpoint.

Data Type

string

Default Value

""

Remarks

The URL of the SOAP Web service endpoint is provided by Marketo on the Admin page of the Marketo website.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer

The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Bulk API

Specifies whether to use the Marketo Bulk API.

Data Type

bool

Default Value

false

Remarks

When set to 'True', the Marketo Bulk API will be used to extract or import data, where applicable. The Bulk API in an interface that allows you to retrieve or import large sets of data using delimited (CSV, TSV, or SSV) files. Currently the only tables that support the Bulk API are: Leads (extract and import) and Activities (extract). For any tables that do not support the Bulk API, this property will be ignored.

The Bulk API causes all the data to be retrieved in a single request and requires the data to be accumulated on the server side prior to sending. Therefore requesting a large amount of data using the Bulk API may be advantageous over using the REST API and you may see performance improvements. Additionally the Bulk API requires less API requests to be made (which helps preserve your API calls and staying within the API restrictions enforced by Marketo).

To use the Bulk API to extract records, a job must be created and enqueued. Once enqueued, Marketo will begin processing the job to retrieve the requested data and generate the delimited file. The status of the job can be polled to determine the current status and whether the file is available to be downloaded. Once the status shows that the job is complete and the file is ready, the data can then be downloaded.

When UseBulkAPI is set to True and JobPollingInterval is set to a value greater than 0, the adapter will perform all the previous mentioned steps for you when executing a SELECT query on a Leads or Activities table. This will create and enqueue a job with the specified columns and filters. Note that a filter is required when exporting bulk data. For the Activities tables, an ActivityDate range must be specified. For the Leads table, a CreatedAt or UpdatedAt range may be specified or a Static or Smart list. The adapter will poll the job status to identify when the job has completed, waiting JobPollingInterval seconds in between calls. Once the job is complete, the adapter will download the delimited file that was created, parse it, and return the results for the specified query.

Note that job status calls count against your API call limit and thus it is suggested to space out your status requests based on the amount of data you are requesting. The job status polling interval is configurable via JobPollingInterval. Marketo will only update the status every 60 seconds and thus it is suggested that your polling interval be larger than 60 seconds. When expecting large datasets, it may be best to increase the polling interval to a value greater than 5 minutes to minimize API calls. It is possible that it may take a while for the job to be processed and thus it may seem like the query is exhibiting a hanging behavior when it is actually just waiting for the job to complete.

In the case that you want to issue your own job status polling requests, you can set JobPollingInterval to 0. This will just create and enqueue the job for you when you execute a SELECT query on a Leads or Activities table, returning the JobId in the result set.

Once a job has been enqueued, the status of the job can be polled by calling the GetExportJobStatus stored procedure.

The JobStatus value will be 'Complete' signaling that the job has finished processing and is ready to be downloaded. To finish executing your initial SELECT query, add the JobId filter to the WHERE clause of the initial SELECT statement. This query will download the file for the specified JobId and parse the result set.

Logic/Code Example (JobPollingInterval = 0):

```
SELECT JobId, Company, FirstName AS fn, LastName AS ln FROM Leads
WHERE CreatedAt>='10/01/2017' AND CreatedAt<'10/31/2017'
# Retrieve the JobId value from the ResultSet (e.g.
c4ebf745-b0e3-4bb8-bfc9-bd8472a28d35). Only one row is returned and
JobId will be the only relevant value returned.

loop(desired time interval) {
  EXEC GetExportJobStatus
@JobId='c4ebf745-b0e3-4bb8-bfc9-bd8472a28d35', @Type='Leads'
  if (JobStatus == 'Completed') break;
}

SELECT Company, FirstName AS fn, LastName AS ln FROM Leads WHERE
CreatedAt>='10/01/2016' AND CreatedAt<'10/31/2016' AND
JobId='c4ebf745-b0e3-4bb8-bfc9-bd8472a28d35'
```

Note: this property is only applicable when using the REST API.

User Id

The Marketo SOAP API User Id.

Data Type

string

Default Value

""

Remarks

The User Id is provided by Marketo and is used to authenticate to the Marketo SOAP Web service.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Marketo Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to the REST API

To connect to Marketo using the REST API the OAuthClientId, OAuthClientSecret, and RESTEndpoint properties, under the OAuth and REST Connection sections, must be set to valid Marketo user credentials.

OAuth Authentication

OAuth Authentication with Marketo requires a custom service with REST API access.

Creating a Custom Service

A custom service is required to connect to Marketo using the REST API. The below steps will guide you in creating a custom service.

Navigate to the admin area of your Marketo application.

Click Users & Roles in the Security section.

Select the Roles tab and click New Role to create a new Role.

Enter a Role Name and select the permissions for the Role. The Access API permissions are specific to the REST API.

Now that an API Role is created, select the Users tab and click Invite New User.

Enter the new user information and select the role that was just created with API access. The API Only option can be selected to denote the user as an API Only user.

Now that a new user has been created, a new service will need to be created. Click the LaunchPoint option (Admin -> Integration -> LaunchPoint).

Click New Service.

Select the Custom service type and enter a display name and description.

Select the user you created.

Obtaining the OAuthClientId and OAuthClientSecret Values

To obtain the OAuthClientId and OAuthClientSecret, navigate to the LaunchPoint option on the Admin area. Click the View Details link for the desired service. A window containing the authentication credentials is displayed.

Obtaining the REST Endpoint URL

The RESTEndpoint can be found on your Marketo Admin area on the Integration -> Web Services option in the REST API section. Note the Identity Endpoint will not be needed.

Connecting to the SOAP API

To connect to Marketo using the SOAP API the UserId, EncryptionKey, and SOAPEndpoint properties, under the SOAP Connection section, must be set to valid Marketo user credentials.

These values can be found on your Marketo Admin area in the Integration -> Web Services option under the SOAP API section.

SQL Compliance

The Marketo Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See REST Data Model for information on the capabilities of the Marketo API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

UPSERT Statements

An UPSERT updates a record if it exists and inserts the record if it does not. See UPSERT Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Marketo adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | AND |
OR | IN } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Leads
```

Rename a column:

```
SELECT "Email" AS MY_Email FROM Leads
```

Search data:

```
SELECT * FROM Leads WHERE Email = 'brucew@waynetech.com';
```

The Marketo APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, AND, OR, IN.

```
SELECT * FROM Leads WHERE Email = 'brucew@waynetech.com';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Email INTO "csv://Leads.txt" FROM "Leads" WHERE Email =
'brucew@waynetech.com'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Email INTO "csv://Leads.txt;delimiter=tab" FROM "Leads"
WHERE Email = 'brucew@waynetech.com'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Leads (Email) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
```

```
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Leads SET Email='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "c2ef66a5-a545-413b-9312-79a53caadbc4");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

UPSERT Statements

An UPSERT statement will update an existing record or create a new record if an existing record is not identified.

UPSERT Syntax

The UPSERT syntax is the same as for insert. Marketo uses the input provided in the VALUES clause to determine whether the record already exists. If the record does not exist, all columns required to insert the record must be specified. See REST Data Model for any table-specific information.

```
UPSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "UPSERT INTO Leads (Email) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:marketo:Schema=REST;RESTEndpoint
=https://MyMarketoUrl/rest;OAuthClientId=MyOAuthClientId;OAuthClie
ntSecret=MyOAuthClientSecret;",);
String cmd = "DELETE FROM Leads WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "c2ef66a5-a545-413b-9312-79a53caadbc4");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

SOAP Data Model

The Marketo Adapter models Marketo entities in relational Tables, Views, and Stored Procedures. API limitations and requirements are documented in the following sections; you can use the SupportEnhancedSQL feature, set by default, to circumvent most of these limitations.

A full list is available upon customer request.

TDV MongoDB Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to MongoDB

Set the following connection properties to connect to a single MongoDB instance:

Server: Set this to the name or address of the server your MongoDB instance is running on. You can specify the port here or in Port.

Database: Set this to the database you want to read from and write to.

Connecting to Replica Sets

To connect to a replica set, set the following in addition to the preceding connection properties:

ReplicaSet: Set this to a comma-separated list of secondary servers in the replica set, specified by address and port.

SlaveOK: Set this to true if you want to read from secondary (slave) servers.

ReadPreference: Set this to fine-tune how the adapter reads from secondary servers.

Securing MongoDB Connections

You can set UseSSL to negotiate SSL/TLS encryption when you connect.

Authenticating MongoDB Connections

Supported authentication types are challenge-response authentication and LDAP. To authenticate to MongoDB, select an authentication type in AuthMechanism and set User and Password. If you need to work with data in one database and authenticate to another database, set AuthDatabase in addition to Database.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Database	The name of the MongoDB database for authentication.
Auth Mechanism	The authentication mechanism that MongoDB will use to authenticate the connection.
Database	The name of the MongoDB database.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Flatten Arrays	By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.
Flatten Objects	Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.
Generate Schema Files	Indicates the user preference as to when schemas should be generated and saved.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password used to authenticate with MongoDB.
Port	The port for the MongoDB database.
Query Passthrough	This option passes the query to MongoDB as-is.
Readonly	You can use this property to enforce read-only access to MongoDB from the provider.
Read Preference	Set this to a strategy for reading from a replica set. Accepted values are primary, primaryPreferred, secondary, secondaryPreferred, and nearest.
Replica Set	This property allows you to specify multiple servers in addition to the one configured in Server and Port . Specify both a server name and port; separate servers with a comma.
Row Scan Depth	The maximum number of rows to scan to look for the columns available in a table. Set this property to gain more control over how the provider applies data types to collections.
Server	The host name or IP address of the server hosting the MongoDB database.
Slave OK	This property sets whether the provider is allowed to read from secondary (slave) servers.
SSL Client Cert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSL Client Cert Password	The password for the TLS/SSL client certificate.
SSL Client Cert Subject	The subject of the TLS/SSL client certificate.
SSL Client Cert Type	The type of key store containing the TLS/SSL client certificate.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.

Type Detection Scheme	Comma-separated options for how the provider will scan the data to determine the fields and datatypes in each document collection.
User	The username used to authenticate with MongoDB.
Use SSL	This field sets whether SSL is enabled.

Auth Database

The name of the MongoDB database for authentication.

Data Type

string

Default Value

""

Remarks

The name of the MongoDB database for authentication. Only needed if the authentication database is different from the database to retrieve data from.

Auth Mechanism

The authentication mechanism that MongoDB will use to authenticate the connection.

Data Type

string

Default Value

"NONE"

Remarks

Accepted values are MONGODB-CR, SCRAM-SHA-1, PLAIN, and NONE. The following authentication types correspond to the authentication values.

Authenticating with Challenge-Response

Generally, this property does not need to be set for this authentication type, as the adapter uses different challenge-response mechanisms by default to authenticate a user to different versions of MongoDB.

MongoDB 2: MongoDB 2 uses MONGODB-CR to authenticate.

MongoDB 3.x: MongoDB 3 uses SCRAM-SHA-1 by default; new users you create in MongoDB 3 use this authentication method. However, MongoDB 3 servers will continue to use MONGODB-CR to authenticate users created in MongoDB 2.6.

The User and Password properties correspond to a username and password stored in a MongoDB database. If you want to connect to data from one database and authenticate to another database, set both Database and AuthDatabase.

Authenticating with LDAP

Set AuthMechanism to PLAIN to use LDAP authentication. Additionally, set AuthDatabase to the database storing the User and Password credentials.

This value specifies the SASL PLAIN mechanism; note that this mechanism transmits credentials over plain-text, so it is not suitable for use without TLS/SSL on untrusted networks. Set UseSSL and SSLServerCert to enable TLS/SSL negotiation with the Server.

Database

The name of the MongoDB database.

Data Type

string

Default Value

""

Remarks

The name of the MongoDB database.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to MongoDB and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.

SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Flatten Arrays

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. Set FlattenArrays to the number of elements you want to return from nested arrays.

Data Type

string

Default Value

""

Remarks

By default, nested arrays are returned as strings of JSON. The FlattenArrays property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

Set FlattenArrays to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

For example, you can return an arbitrary number of elements from an array of strings:

```
["FLOW-MATIC","LISP","COBOL"]
```

When FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
languages.0	FLOW-MATIC

Setting FlattenArrays to -1 will flatten all the elements of nested arrays.

Flatten Objects

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON.

Data Type

bool

Default Value

true

Remarks

Set FlattenObjects to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of JSON. To generate the column name, the adapter concatenates the property name onto the object name with a dot.

For example, you can flatten the nested objects below at connection time:

```
[
     { "grade": "A", "score": 2 },
     { "grade": "A", "score": 6 },
     { "grade": "A", "score": 10 },
     { "grade": "A", "score": 9 },
     { "grade": "B", "score": 14 }
]
```

When FlattenObjects is set to true and FlattenArrays is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
grades.0.grade	A
grades.0.score	2

Generate Schema Files

Indicates the user preference as to when schemas should be generated and saved.

Data Type

string

Default Value

"Never"

Remarks

GenerateSchemaFiles enables you to persist the table definitions identified by Automatic Schema Discovery. This property outputs schemas to .rsd files in the path specified by Location.

Available settings are the following:

Never: A schema file will never be generated.

OnUse: A schema file will be generated the first time a table is referenced, provided the schema file for the table does not already exist.

OnStart: A schema file will be generated at connection time for any tables that do not currently have a schema file.

Note that if you want to regenerate a file, you will first need to delete it.

Generate Schemas with SQL

When you set GenerateSchemaFiles to **OnUse**, the adapter generates schemas as you execute SELECT queries. Schemas are generated for each table referenced in the query.

Generate Schemas on Connection

Another way to use this property is to obtain schemas for every table in your database when you connect. To do so, set GenerateSchemaFiles to **OnStart** and connect.

Editing Schemas

Schema files have a simple format that makes them easy to modify. See Custom Schema Definitions for an end-to-end guide using the MongoDB restaurants collection.

Using Dynamic Schemas Instead

If your data structures are volatile, consider setting GenerateSchemaFiles to **Never** and using dynamic schemas, which change based on the metadata retrieved when you connect. Also, note that you cannot execute Free-Form Queries queries to a table that has a static schema definition.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password used to authenticate with MongoDB.

Data Type

string

Default Value

""

Remarks

The password used to authenticate with MongoDB.

Port

The port for the MongoDB database.

Data Type

string

Default Value

"27017"

Remarks

The port for the MongoDB database.

Query Passthrough

This option passes the query to MongoDB as-is.

Data Type

bool

Default Value

false

Remarks

When set to 'True', the specified query will be passed to MongoDB as-is. The supported query syntax are the Mongo Shell Methods. Currently only the 'db.collection.find()' method is supported. Thus the same query that you would enter in a Mongo Shell can be issued here when QueryPassthrough is set to 'True'.

Note that you can use the EVAL stored procedure to execute other JavaScript functions.

Readonly

You can use this property to enforce read-only access to MongoDB from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Read Preference

Set this to a strategy for reading from a replica set. Accepted values are primary, primaryPreferred, secondary, secondaryPreferred, and nearest.

Data Type

string

Default Value

""

Remarks

This property enables you to execute queries to a member in a replica set other other than the primary member. Accepted values are the following:

primary: All SELECT queries are executed against the primary server.

primaryPreferred: If the primary server is not available, SELECT queries are executed to a secondary server.

secondary: All SELECT queries are executed to the secondary servers.

secondaryPreferred: SELECT queries are executed to a secondary server if one is available. Otherwise, the queries are executed to the primary server.

nearest: SELECT queries are executed to the server with the least latency.

When to Use ReadPreference

When this property is set, query results may not reflect the latest changes if a write operation has not yet been replicated to a secondary machine. You can use ReadPreference to accomplish the following, with some risk that the adapter will return stale data:

Configure failover queries: If the primary server is unavailable, you can set this property to "primaryPreferred" to continue to execute queries online.

Execute faster queries to geographically distributed replica sets: If your deployment uses multiple data centers, setting ReadPreference to "nearest" can result in faster queries, as the adapter executes SELECT queries to whichever replica set member has the lowest latency.

When directing the adapter to execute SELECT statements to a secondary server, SlaveOK must also be set. Otherwise, the adapter will return an error response.

Replica Set

This property allows you to specify multiple servers in addition to the one configured in Server and Port . Specify both a server name and port; separate servers with a comma.

Data Type

string

Default Value

""

Remarks

To work with a replica set, you must specify all servers in the replica set in this property and Server and Port. Specify both a server name and port; separate servers with a comma. For example:

```
Server=localhost;Port=27017;ReplicaSet=localhost:27018,
localhost:27019;
```

To find the primary the adapter queries the servers in ReplicaSet and the server specified by Server and Port.

Note that only the primary in a replica set is writable. Secondaries can be readable if the SlaveOK setting allows it. To configure a strategy executing SELECT queries to secondaries, see ReadPreference.

Row Scan Depth

The maximum number of rows to scan to look for the columns available in a table. Set this property to gain more control over how the provider applies data types to collections.

Data Type

string

Default Value

"100"

Remarks

Since MongoDB is schemaless, the columns in a table must be determined by scanning table rows. This value determines the maximum number of rows that will be scanned. The default value is 100.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

Server

The host name or IP address of the server hosting the MongoDB database.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the server hosting the MongoDB database.

Slave OK

This property sets whether the provider is allowed to read from secondary (slave) servers.

Data Type

bool

Default Value

false

Remarks

This property sets whether the adapter is allowed to read from secondary (slave) servers in a replica set. You can fine-tune how the adapter queries secondary servers with ReadPreference.

SSL Client Cert

The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The SSLClientCertType field specifies the type of the certificate store specified by SSLClientCert. If the store is password protected, specify the password in SSLClientCertPassword.

SSLClientCert is used in conjunction with the SSLClientCertSubject field in order to specify client certificates. If SSLClientCert has a value, and SSLClientCertSubject is set, a search for a certificate is initiated. Please refer to the SSLClientCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

SSL Client Cert Password

The password for the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

SSL Client Cert Subject

The subject of the TLS/SSL client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State

C	Country
E	Email Address

If a field value contains a comma it must be quoted.

SSL Client Cert Type

The type of key store containing the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.

JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Type Detection Scheme

Comma-separated options for how the provider will scan the data to determine the fields and datatypes in each document collection.

Data Type

string

Default Value

"RowScan,Recent"

Remarks

None	Setting TypeDetectionScheme to None will return all columns as a string type. Cannot be combined with other options.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned. Can be used with Recent.
Recent	Setting TypeDetectionScheme to Recent will determine whether RowScan is executed on the most recent documents in the collection. Can be used with RowScan.

User

The username used to authenticate with MongoDB.

Data Type

string

Default Value

""

Remarks

The username used to authenticate with MongoDB.

Use SSL

This field sets whether SSL is enabled.

Data Type

bool

Default Value

false

Remarks

This field sets whether the adapter will attempt to negotiate TLS/SSL connections to the server. By default, the adapter checks the server's certificate against the system's trusted certificate store. To specify another certificate, set SSLServerCert.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the MongoDB Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Accessing NoSQL Tables

The adapter implements Automatic Schema Discovery that is highly configurable. The following sections outline the adapter's defaults and link to ways to further customize.

Flattening Nested JSON

By default, the adapter projects columns over the properties of objects. Arrays are returned as JSON strings, by default. You can use the following properties to access array elements, including objects nested in arrays.

FlattenArrays: Set this property to the number of array elements that you want to return as column values. You can also use this property with FlattenObjects to extract the properties of objects nested in arrays.

FlattenObjects: By default, this is true; that is, the properties of objects and nested objects are returned as columns. When you set FlattenArrays, objects nested in the specified array elements are also flattened and returned as columns.

Other mechanisms for accessing nested objects are detailed in NoSQL Database.

Fine-Tuning Data Access

You can use the following properties to gain greater control over MongoDB API features and the strategies the adapter uses to surface them:

RowScanDepth: This property determines the number of rows that will be scanned to detect column data types when generating table metadata.

TypeDetectionScheme: This property allows more control over the strategy implemented by the RowScanDepth property.

QueryPassthrough: This property enables you to execute MongoDB queries through the adapter. See Query Mapping

for JavaScript examples and the corresponding SQL.

GenerateSchemaFiles: This property enables you to persist table metadata in static schema files that are easy to customize, to persist your changes to column data types, for example. You can set this property to "OnStart" to generate schema files for all tables in your database at connection. Or, you can generate schemas as you execute SELECT queries to tables. The resulting schemas are based on the connection properties you use to configure Automatic Schema Discovery

.

To use the resulting schema files, set the Location property to the folder containing the schemas.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

To connect set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate specify FirewallUser and FirewallPassword. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain).

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

NoSQL Database

MongoDB is a schemaless, document database that provides high performance, availability, and scalability. These features are not necessarily incompatible with a standards-compliant query language like SQL-92. In this section we will show various schemes that the adapter offers to bridge the gap with relational SQL and a document database.

Working with MongoDB Objects as Tables

The adapter models the schemaless MongoDB objects into relational tables and translates SQL queries into MongoDB queries to get the requested data. See Query Mapping for more details on how various MongoDB operations are represented as SQL.

Discovering Schemas Automatically

The Automatic Schema Discovery scheme automatically finds the data types in a MongoDB object by scanning a configured number of rows of the object. You can use RowScanDepth, FlattenArrays, and FlattenObjects to control the relational representation of the collections in MongoDB. You can also write Free-Form Queries not tied to the schema.

Customizing Schemas

Optionally, you can use Custom Schema Definitions to project your chosen relational structure on top of a MongoDB object. This allows you to define your chosen names of columns, their data types, and the location of their values in the collection.

Set GenerateSchemaFiles to save the detected schemas as simple configuration files that are easy to extend. You can persist schemas for all collections in the database or for the results of SELECT queries.

Automatic Schema Discovery

The adapter automatically infers a relational schema by inspecting a series of MongoDB documents in a collection. You can use the RowScanDepth property to define the number of documents the adapter will scan to do so. The columns identified during the discovery process depend on the FlattenArrays and FlattenObjects properties.

If FlattenObjects is set, all nested objects will be flattened into a series of columns. For example, consider the following document:

```
{
  id: 12,
  name: "Lohia Manufacturers Inc.",
  address: {street: "Main Street", city: "Chapel Hill", state:
"NC"},
  offices: ["Chapel Hill", "London", "New York"],
  annual_revenue: 35,600,000
}
```

This document will be represented by the following columns:

Column Name	Data Type	Example Value
id	Integer	12
name	String	Lohia Manufacturers Inc.
address.street	String	Main Street
address.city	String	Chapel Hill
address.state	String	NC
offices	String	["Chapel Hill", "London", "New York"]
annual_revenue	Double	35,600,000

If FlattenObjects is not set, then the address.street, address.city, and address.state columns will not be broken apart. The address column of type string will instead represent the entire object. Its value would be {street: "Main Street", city: "Chapel Hill", state: "NC"}. See JSON Functions for more details on working with JSON aggregates.

The FlattenArrays property can be used to flatten array values into columns of their own. This is only recommended for arrays that are expected to be short, for example the coordinates below:

```
"coord": [ -73.856077, 40.848447 ]
```

The FlattenArrays property can be set to 2 to represent the array above as follows:

Column Name	Data Type	Example Value
coord.0	Float	-73.856077
coord.1	Float	40.848447

It is best to leave other unbounded arrays as they are and piece out the data for them as needed using JSON Functions.

Free-Form Queries

As discussed in Automatic Schema Discovery, intuited table schemas enable SQL access to unstructured MongoDB data. JSON Functions enable you to use standard JSON functions to summarize MongoDB data and extract values from any nested structures. Custom Schema Definitions enable you to define static tables and give you more granular control over the relational view of your data; for example, you can write schemas defining parent/child tables or fact/dimension tables. However, you are not limited to these schemes.

After connecting you can query any nested structure without flattening the data. Any relations that you can access with FlattenArrays and FlattenObjects can also be accessed with an ad hoc SQL query.

Let's consider an example document from the MongoDB Restaurant data set:

```
{
  "address": {
    "building": "1007",
    "coord": [
      -73.856077,
      40.848447
    ],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
```

```
  "grades": [
    {
      "grade": "A",
      "score": 2,
      "date": {
        "$date": "1393804800000"
      }
    },
    {
      "date": {
        "$date": "1378857600000"
      },
      "grade": "B",
      "score": 6
    },
    {
      "score": 10,
      "date": {
        "$date": "1358985600000"
      },
      "grade": "C"
    }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

You can access any nested structure in this document as a column. Use the dot notation to drill down to the values you want to access as shown in the query below. Note that arrays have a zero-based index. For example, the following query retrieves the second grade for the restaurant in the example:

```
SELECT "address.building", "grades.1.grade" FROM restaurants WHERE
restaurant_id = '30075445'
```

The preceding query returns the following results:

Column Name	Data Type	Example Value
address.building	String	1007
grades.1.grade	String	A

Vertical Flattening

It is possible to retrieve an array of documents as if it were a separate table. Take the following JSON structure from the restaurants collection for example:

```
{
  "_id" : ObjectId("568c37b748ddf53c5ed98932"),
  "address" : {
    "building" : "1007",
    "coord" : [-73.856077, 40.848447],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [{
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    }, {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    }, {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    }, {
      "date" : ISODate("2011-11-23T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    }, {
      "date" : ISODate("2011-03-10T00:00:00Z"),
      "grade" : "B",
      "score" : 14
    }],
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
```

Vertical flattening will allow you to retrieve the grades array as a separate table:

```
SELECT * FROM "restaurants.grades"
```

This query returns the following data set:

date	grade	score	P_id	_index
2014-03-03T00:00:00.000Z	A	2	568c37b748ddf53c5ed98932	1
2013-09-11T00:00:00.000Z	A	6	568c37b748ddf53c5ed98932	2

2013-01-24T00:00:00.000Z	A	10	568c37b748ddf53c5ed98932	3

You may also want to include information from the base restaurants table. You can do this with a join. Flattened arrays can only be joined with the root document. The adapter expects the left part of the join is the array document you want to flatten vertically. Disable SupportEnhancedSQL to join nested MongoDB documents -- this type of query is supported through the MongoDB API.

```
SELECT "restaurants"."restaurant_id", "restaurants.grades".* FROM
"restaurants.grades" JOIN "restaurants" WHERE "restaurants".name =
'Morris Park Bake Shop'
```

This query returns the following data set:

restaurant_id	date	grade	score	P_id	_index
30075445	2014-03-03T00:00:00.000Z	A	2	568c37b748ddf53c5ed98932	1
30075445	2013-09-11T00:00:00.000Z	A	6	568c37b748ddf53c5ed98932	2
30075445	2013-01-24T00:00:00.000Z	A	10	568c37b748ddf53c5ed98932	3
30075445	2011-11-23T00:00:00.000Z	A	9	568c37b748ddf53c5ed98932	4
30075445	2011-03-10T00:00:00.000Z	B	14	568c37b748ddf53c5ed98932	5

JSON Functions

The adapter can return JSON structures as column values. The adapter enables you to use standard SQL functions to work with these JSON structures. The examples in this section use the following array:

```
[
     { "grade": "A", "score": 2 },
     { "grade": "A", "score": 6 },
     { "grade": "A", "score": 10 },
     { "grade": "A", "score": 9 },
     { "grade": "B", "score": 14 }
]
```

JSON_EXTRACT

The JSON_EXTRACT function can extract individual values from a JSON object. The following query returns the values shown below based on the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_EXTRACT(grades,'[0].grade') AS Grade,
JSON_EXTRACT(grades,'[0].score') AS Score FROM Students;
```

Column Name	Example Value
Grade	A
Score	2

JSON_COUNT

The JSON_COUNT function returns the number of elements in a JSON array within a JSON object. The following query returns the number of elements specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_COUNT(grades,'[x]') AS NumberOfGrades FROM
Students;
```

Column Name	Example Value
NumberOfGrades	5

JSON_SUM

The JSON_SUM function returns the sum of the numeric values of a JSON array within a JSON object. The following query returns the total of the values specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_SUM(score,'[x].score') AS TotalScore FROM
Students;
```

Column Name	Example Value

TotalScore	41

JSON_MIN

The JSON_MIN function returns the lowest numeric value of a JSON array within a JSON object. The following query returns the minimum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MIN(score,'[x].score') AS LowestScore FROM
Students;
```

Column Name	Example Value
LowestScore	2

JSON_MAX

The JSON_MAX function returns the highest numeric value of a JSON array within a JSON object. The following query returns the maximum value specified by the JSON path passed as the second argument to the function:

```
SELECT Name, JSON_MAX(score,'[x].score') AS HighestScore FROM
Students;
```

Column Name	Example Value
HighestScore	14

DOCUMENT

The DOCUMENT function can be used to retrieve the entire document as a JSON string. See the following query and its result as an example:

```
SELECT DOCUMENT(*) from Customers;
```

The query above will return the entire document as shown.

```
{ "id": 12, "name": "Lohia Manufacturers Inc.", "address": {
"street": "Main Street", "city": "Chapel Hill", "state": "NC"},
```

```
"offices": [ "Chapel Hill", "London", "New York" ],
"annual_revenue": 35,600,000 }
```

Query Mapping

The adapter maps SQL queries into the corresponding MongoDB queries. A detailed description of all the transformations is out of scope, but we will describe some of the common elements that are used. The adapter takes advantage of MongoDB features such as the aggregation framework to compute the desired results.

SELECT Queries

The SELECT statement is mapped to the find() function as shown below:

SQL Query	MongoDB Query
`SELECT * FROM Users`	`db.users.find()`

`SELECT user_id, status`
`FROM Users`
	`db.users.find(`
	`  {},`
	`  { user_id: 1, status: 1, _id: 0 }`
	`)`

`SELECT *`
`FROM Users`
`WHERE status = 'A'`
	`db.users.find(`
	`  { status: "A" }`
	`)`

`SELECT *`
`FROM Users`
`WHERE status = 'A' OR age=50`
	`db.users.find(`
	`  { $or: [ { status: "A" },`
	`  { age: 50 } ] }`
	`)`

`SELECT *`
`FROM Users`
`WHERE name LIKE 'A%'`
	`db.users.find(`
	`  {name: /^a/}`
	`)`

`SELECT * FROM Users`
`WHERE status = 'A'`
`ORDER BY user_id ASC`
	`db.users.find( { status: "A" }.sort( { user_id: 1`
	`} )`

`SELECT *`
`FROM Users`
`WHERE status = 'A'`
`ORDER BY user_id DESC`
	`db.users.find( {status: "A" }.sort( {user_id: -1}`
	`)`

Aggregate Queries

The MongoDB aggregation framework was added in MongoDB version 2.2. The adapter makes extensive use of this for various aggregate queries. See some examples below:

SQL Query	MongoDB Query

`SELECT Count(*) As Count`
`FROM Orders`
	`db.orders.aggregate( [`
	`  {`
	`  $group: {`
	`  _id: null,`
	`  count: { $sum: 1 }`
	`  }`
	`  }`
	`] )`

`SELECT Sum(price) As Total`
`FROM Orders`
	`db.orders.aggregate( [`
	`  {`
	`  $group: {`
	`  _id: null,`
	`  total: { $sum: "$price" }`
	`  }`
	`  }`
	`] )`

```
SELECT cust_id, Sum(price) As total
FROM Orders
GROUP BY cust_id
ORDER BY total
```

```
db.orders.aggregate( [
  {
    $group: {
      _id: "$cust_id",
      total: { $sum: "$price" }
    }
  } ,
  { $sort: {total: 1 } }
] )
```

```
SELECT cust_id, ord_date,
Sum(price) As total
FROM Orders
GROUP BY cust_id, ord_date
HAVING total > 250
```

```
db.orders.aggregate( [
  {
    $group: {
      _id: {
        cust_id: "$cust_id",
        ord_date: {
          month: { $month:
"$ord_date" },
          day: { $dayOfMonth:
"$ord_date" },
          year: { $year:
"$ord_date"}
        }
      },
      total: { $sum: "$price" }
    }
  },
  { $match: { total: { $gt: 250 } }
}
] )
```

Insert Statements

The INSERT statement is mapped to the insert function as shown below:

SQL Query	MongoDB Query
`INSERT INTO users(user_id, age, status, [address.city], [address.postalcode]) VALUES ('bcd001', 45, 'A', 'Chapel Hill', 27517)`	`db.users.insert( { user_id: "bcd001", age: 45, status: "A", address:{ city:"Chapel Hill", postalCode:27514} } )`

Update Statements

The UPDATE statement is mapped to the update function as shown below:

SQL Query	MongoDB Query

`UPDATE users`
`SET status = 'C',`
`[address.postalcode] = 90210`
`WHERE age > 25`
	`db.users.update(`
	`  { age: { $gt: 25 } },`
	`  { $set: { status: "C",`
	`address.postalCode: 90210 },`
	`  { multi: true }`
	`)`

Delete Statements

The DELETE statement is mapped to the delete function as shown below:

SQL Query	MongoDB Query
`DELETE FROM users WHERE status = 'D'`	`db.users.remove( { status: "D" } )`

Custom Schema Definitions

You can extend the table schemas created with Automatic Schema Discovery by saving them into schema files. The schema files have a simple format that makes the schemas to edit.

Generating Schema Files

Set GenerateSchemaFiles to "OnStart" to persist schemas for all tables when you connect. You can also generate table schemas as needed: Set GenerateSchemaFiles to "OnUse" and execute a SELECT query to the table.

For example, consider a schema for the restaurants data set. This is a sample data set provided by MongoDB. To download the data set, follow the Getting Started with MongoDB guide.

Below is an example document from the collection:

```
{
  "address":{
    "building":"461",
      "coord":[
        -74.138492,
        40.631136
      ],
      "street":"Port Richmond Ave",
      "zipcode":"10302"
   },
   "borough":"Staten Island",
   "cuisine":"Other",
   "name":"Indian Oven",
   "restaurant_id":"50018994"
}
```

Importing the MongoBD Restaurant Data Set

You can use the mongoimport utility to import the data set:

```
mongoimport --db test --collection restaurants --drop --file data
set.json
```

Customizing a Schema

When GenerateSchemaFiles is set, the adapter saves schemas into the folder specified by the Location property. You can then change column behavior in the resulting schema.

The following schema uses the *other:bsonpath* property to define where in the collection to retrieve the data for a particular column. Using this model you can flatten arbitrary levels of hierarchy.

The *collection* attribute specifies the collection to parse. The *collection* attribute gives you the flexibility to use multiple schemas for the same collection. If *collection* is not specified, the filename determines the collection that is parsed.

Below are the column definitions and the collection to extract the column values from. In Custom Schema Example, you will find the complete schema.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">
```

```
  <rsb:info title="StaticRestaurants" description="Custom Schema
for the MongoDB restaurants data set.">
    <!-- Column definitions -->
    <attr  name="borough"   xs:type="string"
other:bsonpath="$.borough"              />
    <attr  name="cuisine"   xs:type="string"
other:bsonpath="$.cuisine"              />
    <attr  name="building"  xs:type="string"
other:bsonpath="$.address.building"     />
    <attr  name="street"    xs:type="string"
other:bsonpath="$.address.street"       />
    <attr  name="latitude"  xs:type="double"
other:bsonpath="$.address.coord.0"      />
    <attr  name="longitude" xs:type="double"
other:bsonpath="$.address.coord.1"      />
    <input name="rows@next" desc="Internal attribute used for
paging through data."  />
  </rsb:info>

  <rsb:set attr="collection" value="restaurants"/>

</rsb:script>
```

Custom Schema Example

In this section is a complete schema. The *info* section enables a relational view of a MongoDB object. For more details, see Custom Schema Definitions. The table below allows the SELECT, INSERT, UPDATE, and DELETE commands as implemented in the GET, POST, MERGE, and DELETE sections of the schema below.

Use the *collection* attribute to specify the name of the collection you want to parse. You can use the *collection* attribute to define multiple schemas for the same collection.

If *collection* is not specified, the filename determines the collection that is parsed.

Copy the *rows@next* input as-is into your schema. The operations, such as *mongodbadoExecuteSelect*, are internal implementations and can also be copied as is.

```
<rsb:script xmlns:rsb="http://www.rssbus.com/ns/rsbscript/2">

  <rsb:info title="StaticRestaurants" description="Custom Schema
for the MongoDB restaurants data set.">
    <!-- Column definitions -->
    <attr  name="borough"   xs:type="string"
other:bsonpath="$.borough"              />
    <attr  name="cuisine"   xs:type="string"
other:bsonpath="$.cuisine"              />
```

```
    <attr  name="building"  xs:type="string"
other:bsonpath="$.address.building"     />
    <attr  name="street"    xs:type="string"
other:bsonpath="$.address.street"       />
    <attr  name="latitude"  xs:type="double"
other:bsonpath="$.address.coord.0"      />
    <attr  name="longitude" xs:type="double"
other:bsonpath="$.address.coord.1"      />
    <input name="rows@next" desc="Internal attribute used for
paging through data."  />
  </rsb:info>

  <rsb:set attr="collection" value="restaurants"/>

  <rsb:script method="GET">
    <rsb:call op="mongodbadoExecuteSelect">
      <rsb:push />
    </rsb:call>
  </rsb:script>

  <rsb:script method="POST">
    <rsb:call op="mongodbadoExecutePost">
      <rsb:push />
    </rsb:call>
  </rsb:script>

  <rsb:script method="MERGE">
    <rsb:call op="mongodbadoExecuteMerge">
      <rsb:push />
    </rsb:call>
  </rsb:script>

  <rsb:script method="DELETE">
    <rsb:call op="mongodbadoExecuteDelete">
      <rsb:push />
    </rsb:call>
  </rsb:script>

</rsb:script>
```

System Tables

Query the following system tables to access schema information, information on data source functionality, and batch operation statistics.

Schema Tables

The following tables return database metadata for MongoDB:

sys_catalogs

: Lists the available databases.

sys_schemas

: Lists the available schemas.

sys_tables

: Lists the available tables.

sys_tablecolumns

: Describes the columns of the available tables.

sys_views

: Lists the available views.

sys_viewcolumns

: Describes the columns of available views.

sys_procedures

: Describes the available stored procedures.

sys_procedureparameters

: Describes stored procedure parameters.

sys_keycolumns

: Describes the primary and foreign keys.

sys_indexes

: Describes the available indexes.

Data Source Tables

The following tables return information about how to connect to and query the data source:

sys_connection_props

: Returns information on the available connection properties.

sys_sqlinfo

: Describes the SELECT queries that the adapter can offload to the data source.

Query Information Tables

Query statistics are available for data modification queries, including batch operations.

sys_identity

: Returns information about batch operations or single updates.

sys_catalogs

Lists the available databases.

The following query retrieves all databases determined by the connection string:

```
SELECT * FROM sys_catalogs
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.

sys_schemas

Lists the available schemas.

```
SELECT * FROM sys_schemas
```

Columns

Name	Type	Description
CatalogName	*String*	The database name.
SchemaName	*String*	The schema name.

sys_tables

Lists the available tables.

```
SELECT * FROM sys_tables
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table.
TableType	*String*	The table type.
Description	*String*	A description of the table.

sys_tablecolumns

Describes the columns of the available tables.

The following query returns the columns and data types for the Customers table:

```
SELECT ColumnName, DataTypeName FROM sys_tablecolumns WHERE
TableName='Customers'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the table.
SchemaName	*String*	The schema containing the table.
TableName	*String*	The name of the table containing the column.
ColumnName	*String*	The column name.
DataTypeName	*String*	The data type name.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.

Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.
IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	A brief description of the column.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_views

Lists the available views.

```
SELECT TableName FROM sys_views
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.

SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
TableType	*String*	The type of the view.
Description	*String*	A description of the view.

sys_viewcolumns

Describes the columns of the available views.

The following query returns the columns and data types for a specified view:

```
SELECT ColumnName, DataTypeName FROM sys_viewolumns WHERE
TableName='MyView'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the view.
SchemaName	*String*	The name of the schema containing the view.
TableName	*String*	The name of the view.
ColumnName	*String*	The name of the column.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type of the column. This value is determined at run time based on the environment.
Length	*Int32*	The length in characters of the column or the numeric precision.
NumericPrecision	*Int32*	The maximum number of digits in numeric data. The column length in characters for character and date-time data.
NumericScale	*Int32*	The column scale or number of digits to the right of the decimal point.

IsNullable	*Boolean*	Whether the column can contain null.
Description	*String*	The column description.
Ordinal	*Int32*	The sequence number of the column.
IsAutoIncrement	*String*	Whether the column value is assigned in fixed increments.
IsGeneratedColumn	*String*	Whether the column is generated.
IsReadOnly	*Boolean*	Whether the column is read-only.
IsKey	*Boolean*	Whether the column is a primary key.
IsHidden	*Boolean*	Whether the column is hidden.

sys_procedures

Lists the available stored procedures.

```
SELECT * FROM sys_procedures
```

Columns

Name	Type	Description
CatalogName	*String*	The database containing the stored procedure.
SchemaName	*String*	The schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure.
Description	*String*	A description of the stored procedure.

sys_procedureparameters

Describes stored procedure parameters.

The following query returns information about all of the input parameters for the EVAL stored procedure:

```
SELECT * FROM sys_procedureparameters WHERE ProcedureName='EVAL'
AND Direction=1 OR Direction=2
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the stored procedure.
SchemaName	*String*	The name of the schema containing the stored procedure.
ProcedureName	*String*	The name of the stored procedure containing the parameter.
ColumnName	*String*	The name of the stored procedure parameter.
Direction	*Int32*	An integer corresponding to the type of the parameter: input (1), input/output (2), or output(4). input/output type parameters can be both input and output parameters.
DataTypeName	*String*	The name of the data type.
DataType	*Int32*	An integer indicating the data type. This value is determined at run time based on the environment.
Length	*Int32*	The number of characters allowed for character data. The number of digits allowed for numeric data.
NumericPrecision	*Int32*	The maximum precision for numeric data. The column length in characters for character and date-time data.
NumbericScale	*Int32*	The number of digits to the right of the decimal point in numeric data.
IsNullable	*Boolean*	Whether the parameter can contain null.
Description	*String*	The description of the parameter.
Ordinal	*Int32*	The index of the parameter.

sys_keycolumns

Describes the primary and foreign keys.

The following query retrieves the primary key for the Customers table:

```
SELECT * FROM sys_keycolumns WHERE IsKey='True' AND
TableName='Customers'
```

Columns

Name	Type	Description
CatalogName	*String*	The name of the database containing the key.
SchemaName	*String*	The name of the schema containing the key.
TableName	*String*	The name of the table containing the key.
ColumnName	*String*	The name of the key column.
IsKey	*Boolean*	Whether the column is a primary key in the table referenced in the TableName table.
IsForeignKey	*Boolean*	Whether the column is a foreign key referenced in the TableName table.
ReferencedCatalogName	*String*	The database containing the primary key.
ReferencedSchemaName	*String*	The schema containing the primary key.
ReferencedTableName	*String*	The table containing the primary key.
ReferencedColumnName	*String*	The column name of the primary key.

sys_indexes

Describes the available indexes. By filtering on indexes, you can write more selective queries with faster query response times.

The following query retrieves all indexes that are not primary keys:

```
SELECT * FROM sys_indexes WHERE IsPrimary='false'
```

Columns

Name	Type	Description

CatalogName	*String*	The name of the database containing the index.
SchemaName	*String*	The name of the schema containing the index.
TableName	*String*	The name of the table containing the index.
IndexName	*String*	The index name.
ColumnName	*String*	The name of the column associated with the index.
IsUnique	*Boolean*	True if the index is unique. False otherwise.
IsPrimary	*Boolean*	True if the index is a primary key. False otherwise.
Type	*Int16*	An integer value corresponding to the index type: statistic (0), clustered (1), hashed (2), or other (3).
SortOrder	*String*	The sort order: A for ascending or D for descending.
OrdinalPosition	*Int16*	The sequence number of the column in the index.

sys_connection_props

Returns information on the available connection properties and those set in the connection string.

The following query retrieves all connection properties that have been set in the connection string or set through a default value:

```
SELECT * FROM sys_connection_props WHERE Value <> ''
```

Columns

Name	Type	Description
Name	*String*	The name of the connection property.
ShortDescription	*String*	A brief description.
Type	*String*	The data type of the connection property.
Default	*String*	The default value if one is not explicitly set.

Values	*String*	A comma-separated list of possible values. A validation error is thrown if another value is specified.
Value	*String*	The value you set or a preconfigured default.
Required	*Boolean*	Whether the property is required to connect.
Category	*String*	The category of the connection property.
IsSessionProperty	*String*	Whether the property is a session property, used to save information about the current connection.

sys_sqlinfo

Describes the SELECT query processing that the adapter can offload to the data source.

Collaborative Query Processing

When working with data sources that do not support SQL-92, you can query the sys_sqlinfo view to determine the query capabilities of the underlying APIs, expressed in SQL syntax. See SQL Compliance for SQL syntax details.

Discovering the Data Source's SELECT Capabilities

Below is an example data set of SQL capabilities. Some aspects of SELECT functionality are returned in a comma-separated list if supported; otherwise, the column contains NO.

Name	**Possible Values**
AGGREGATE_FUNCTIONS	AVG, COUNT, MAX, MIN, SUM, DISTINCT
COUNT	YES
SUPPORTED_OPERATORS	=, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IN, NOT IN, IS NULL, IS NOT NULL, AND, OR
GROUP_BY	NO_RELATION

OUTER_JOINS	YES
OJ_CAPABILITIES	NESTED, LEFT, RIGHT, INNER, NOT_ORDERED, ALL_COMPARISON_OPS
SUBQUERIES	COMPARISON, EXISTS, IN, CORRELATED_SUBQUERIES, QUANTIFIED
STRING_FUNCTIONS	ASCII,CHAR,CONCAT,LEFT,LTRIM,REPLACE,RIGHT,RTRIM,SOUNDEX,SPACE,SUBSTRING
NUMERIC_FUNCTIONS	ABS,ACOS,ASIN,ATAN,CEILING,COS,COT,DEGREES,EXP,FLOOR,LOG,LOG10,PI,POWER,RADIANS,RAND,ROUND,SIGN,SIN,SQRT,TAN
TIMEDATE_FUNCTIONS	CURRENT_DATE,CURRENT_TIMESTAMP,MONTH,YEAR
IDENTIFIER_QUOTE_OPEN_CHAR	[
IDENTIFIER_QUOTE_CLOSE_CHAR	]

The following query retrieves the operators that can be used in the WHERE clause:

```
SELECT * FROM sys_sqlinfo WHERE Name='SUPPORTED_OPERATORS'
```

Note that individual tables may have different limitations or requirements on the WHERE clause; refer to the NoSQL Database section for more information.

Columns

Name	Type	Description
NAME	*String*	A component of SQL syntax, or a capability that can be processed on the server.
VALUE	*String*	Detail on the supported SQL or SQL syntax.

sys_identity

Returns information about attempted modifications. The following query retrieves the Ids of the modified rows in a batch operation.

```
SELECT * FROM sys_identity
```

Columns

Name	Type	Description
Id	*String*	The database-generated Id returned from a data modification operation.
Batch	*String*	An identifier for the batch. 1 for a single operation.
Operation	*String*	The result of the operation in the batch: INSERTED, UPDATED, or DELETED.
Message	*String*	SUCCESS or an error message if the update in the batch failed.

Stored Procedures

Stored Procedures are available to complement the data available from the NoSQL Database. Sometimes it is necessary to update data available from a view using a stored procedure because the data does not provide for direct, table-like, two-way updates. In these situations, the retrieval of the data is done using the appropriate view or table, while the update is done by calling a stored procedure. Stored procedures take a list of parameters and return back a dataset that contains the collection of tuples that constitute the response.

MongoDB Adapter Stored Procedures

Name	Description
AddDocument	Insert entire JSON documents to MongoDB as-is.
CreateSchema	Creates a schema file for the collection.
Eval	Provides the ability to run JavaScript code on the MongoDB server.
GetDocument	Take a pass-through query to retrieve documents.
SearchDocument	Get the entire document as a string.

AddDocument

Insert entire JSON documents to MongoDB as-is.

Input

Name	Type	Description
Collection	*String*	The collection name to be inserted.
Document	*String*	The JSON document to be inserted.

Output

Name	Type	Description
Success	*String*	Returns true if the operation is successful, else an exception is returned.

CreateSchema

Creates a schema file for the collection.

Input

Name	Type	Description
SchemaName	*String*	The schema of the collection.
TableName	*String*	The name of the collection.
FileName	*String*	The full name of the generated schema.

Output

Name	Type	Description
Result	*String*	Returns Success or Failure.

Eval

Provides the ability to run JavaScript code on the MongoDB server.

Stored Procedure Specific Information

You can use the EVAL stored procedure to execute JavaScript functions as stored procedures:

```
EXEC EVAL @jsFunction = 'function() { return
db.restaurants.findOne(); }'
```

You can also use EVAL to save functions to system.js

```
EXEC EVAL @jsFunction = 'function() { db.system.js.save({ _id:
"myAddFunction", value : function (x, y) { return x + y; } }); }'
```

And then execute stored JavaScript functions as stored procedures:

```
EXEC EVAL @jsFunction = 'function() { return myAddFunction(1,1); }'
```

You can retrieve a list of all stored JavaScript functions by querying the system.js table:

```
SELECT * FROM "system.js"
```

Input

Name	Type	Description
Jsfunction	*String*	A JavaScript function to execute.

Output

Name	Type	Description
*	*String*	Output will vary for each collection.

GetDocument

Take a pass-through query to retrieve documents.

Input

Name	Type	Description
Collection	*String*	The collection name to be inserted.
Query	*String*	The Mongo pass-through JSON-style query.
Projection	*String*	The Mongo pass-through JSON-style projection.

Output

Name	Type	Description
*	*String*	Output will vary for each collection.

SearchDocument

Get the entire document as a string.

Input

Name	Type	Description
Collection	*String*	The collection name to search.
_id	*String*	The primary key value of the collection.

Output

Name	Type	Description
Document	*String*	Returns the entire document as a string.

SQL Compliance

The MongoDB Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See NoSQL Database for information on the capabilities of the MongoDB API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

UPDATE Statements

The primary key _id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key _id is required to delete a record. See DELETE Statements for a syntax reference and examples.

CREATE TABLE Statements

See CREATE TABLE Statements for a syntax reference and examples.

DROP TABLE Statements

See DROP TABLE Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the MongoDB adapter:

```
SELECT {
  [ TOP <numeric_literal> | DISTINCT ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  } [ , ... ]
  [ [
      INNER
    ] JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [ HAVING <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
    [
      { OFFSET | , }
      <expression>
    ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
     <expression> { = | > | < | >= | <= | <> | != | LIKE | IN | NOT
IN | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Customers
```

Rename a column:
```
SELECT "CompanyName" AS MY_CompanyName FROM Customers
```

Search data:
```
SELECT * FROM Customers WHERE Country = 'US';
```

The MongoDB APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, IN, NOT IN, AND, OR.
```
SELECT * FROM Customers WHERE Country = 'US';
```

Return the number of items matching the query criteria:
```
SELECT COUNT(*) AS MyCount FROM Customers
```

Return the number of unique items matching the query criteria:
```
SELECT COUNT(DISTINCT CompanyName) FROM Customers
```

Return the unique items matching the query criteria:
```
SELECT DISTINCT CompanyName FROM Customers
```

Summarize data:
```
SELECT CompanyName, MAX(Balance) FROM Customers  GROUP BY
CompanyName
```

See Aggregate Functions for details.

Retrieve data from multiple tables.
```
SELECT "restaurants"."restaurant_id", "restaurants".name,
"restaurants.grades".* FROM "restaurants.grades" JOIN
"restaurants" WHERE "restaurants".name = 'Morris Park Bake Shop'
```

See JOIN Queries for details.

Sort a result set in ascending order:
```
SELECT City, CompanyName FROM Customers  ORDER BY CompanyName ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM Customers WHERE Country = 'US'
```

COUNT_DISTINCT

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT_DISTINCT(City) AS DistinctValues FROM Customers WHERE
Country = 'US'
```

COUNT(DISTINCT)

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT(DISTINCT City) AS DistinctValues FROM Customers WHERE
Country = 'US'
```

AVG

Returns the average of the column values.

```
SELECT CompanyName, AVG(Balance) FROM Customers WHERE Country =
'US' GROUP BY CompanyName
```

MIN

Returns the minimum column value.

```
SELECT MIN(Balance), CompanyName FROM Customers WHERE Country =
'US' GROUP BY CompanyName
```

MAX

Returns the maximum column value.

```
SELECT CompanyName, MAX(Balance) FROM Customers WHERE Country =
'US' GROUP BY CompanyName
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(Balance) FROM Customers WHERE Country = 'US'
```

JOIN Queries

The MongoDB Adapter supports joins of a nested array with its parent document.

Joining Nested Structures

The adapter expects the left part of the join is the array document you want to flatten vertically. This type of query is supported through the MongoDB API.

For example, consider the following query from MongoDB's restaurants collection:

```
SELECT "restaurants"."restaurant_id", "restaurants".name,
"restaurants.grades".*
FROM "restaurants.grades"
JOIN "restaurants"
WHERE "restaurants".name = 'Morris Park Bake Shop'
```

See Vertical Flattening for more details.

Projection Functions

JSON_AVG(json, jsonpath)

Computes the average value of a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_COUNT(json, jsonpath)

Returns the number of elements in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MAX(json, jsonpath)

Gets the maximum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MIN(json, jsonpath)

Gets the minimum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_SUM(json, jsonpath)

Computes the sum of the elements in a JSON within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_EXTRACT(json, jsonpath)

Selects any value in a JSON array or object. The path to the array is specified in the jsonpath argument. Return value is numeric or null.

json: The JSON document to extract.

jsonpath: The XPath used to select the nodes. The JSONPath must be a string constant. The values of the nodes selected will be returned in a token-separated list.

XML_EXTRACT(xml, xpath [, separator])

Extracts an XML document using the specified XPath to flatten the XML. A comma is used to separate the outputs by default, but this can be changed by specifying the third parameter.

xml: The XML document to extract.

xpath: The XPath used to select the nodes. The nodes selected will be returned in a token-separated list.

separator: The optional token used to separate the items in the flattened response. If this is not specified, the separator will be a comma.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT City, CompanyName INTO "csv://Customers.txt" FROM
"Customers" WHERE Country = 'US'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT City, CompanyName INTO "csv://Customers.txt;delimiter=tab"
FROM "Customers" WHERE Country = 'US'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO Customers (CompanyName) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "Caterpillar");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { _id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Customers SET CompanyName='Caterpillar' WHERE
_id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "22");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { _id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:mongodb:Server=127.0.0.1;Port=27
017;Database=test;User=test;Password=test;",);
String cmd = "DELETE FROM Customers WHERE _id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "22");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

CREATE TABLE Statements

To create new MongoDB entities, use CREATE TABLE statements:

CREATE TABLE Syntax

The CREATE TABLE statement specifies the table name and a comma-separated list of column names and the primary keys of the table.

```
CREATE TABLE <table_name> IF [ NOT EXISTS ]
(
  {
     <column_name> <data_type>
     [ NOT NULL ]
     [ DEFAULT <literal> ]
     [ PRIMARY KEY ]
     [ UNIQUE ]
  } |  PRIMARY KEY ( <column_name> [ , ... ] )
```

```
  [ , ... ]
)
```

Example Query:

The following statement creates a MyCustomers table on the MongoDB server with name, age, and address columns.

```
CREATE TABLE IF NOT EXISTS "MyCustomers" (name VARCHAR(20), age
INT, address VARCHAR(20))
```

DROP TABLE Statements

Use DROP TABLE statements to delete a table and all the data it contains from MongoDB.

DROP TABLE Syntax

The DROP TABLE statement accepts the name of the table to delete.

```
DROP TABLE [ IF EXISTS ] <table_name>
```

Example Query:

The following query deletes all MyCustomers data from the server.

```
DROP TABLE IF EXISTS MyCustomers
```

Data Model

A full list is available upon customer request.

TDV NetSuite Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

The User and Password properties, under the Authentication section, must be set to valid NetSuite user credentials. In addition, the AccountId must be set to the Id of a company account that can be used by the specified User. The RoleId can be optionally specified to log in the user with limited permissions.

For more information, refer to our Connecting to NetSuite guide.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Account Id	The company account your username is associated with on NetSuite.
Aggregate Column Mode	Indicating how aggregate columns should be treated.
Application Id	As of version 2015.2, requests to NetSuite require an application ID or token-based authentication details.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.

Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Include Child Tables	A boolean indicating if child tables should be displayed.
Include Custom Field Columns	A boolean indicating if you would like to include custom field columns.
Include Custom List Tables	A boolean indicating if you would like to use tables based on custom lists.
Include Custom Record Tables	A boolean indicating if you would like to use tables based on custom record types.
Include Reference Columns	A comma separated list representing the columns to include when retrieving data from a field representing a record reference.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Logout Unknown Sessions	Boolean indicating if unknown sessions already established to the user account should be removed.
Maximum Concurrent Sessions	The maximum number of concurrent sessions available for use by the username specified in the connection.
Netsuite Metadata Folder	A path to a directory to download metadata files from NetSuite. Set this for best performance.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Access Token Secret	The OAuth access token secret for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.

Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Pagesize	The number of results to return per page from NetSuite.
Password	The password of the NetSuite user used to authenticate.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to NetSuite from the provider.
Request Memorized Transactions	A boolean indicating if you would like to request memorized transactions when retrieving transactions from NetSuite.
Role Id	The RoleId is the InternalId of the role that will be used to log in to NetSuite. Leave empty to use the user's default role.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Async Services	A boolean indicating if you would like to use asynchronous services when inserting, updating, and deleting.
User	The user of the NetSuite account used to authenticate.

User Timezone Offset	Your user timezone offset as defined in your NetSuite preferences under Home --> Preferences --> Time Zone. Ex: -8:00.
Use Sandbox	A boolean indicating if you would like to use the sandbox instance of your NetSuite account.
Use Sessions	Establish sessions with NetSuite, or instead submit user credentials on each request.
Use Upserts	A boolean indicating if you would like to perform an upsert when an insert operation is used.
Version	The version of the NetSuite API in usage. Defaults to 2017_1.
Web Service Host	An optional override for the web service host such as webservices.na1.netsuite.com.

Account Id

The company account your username is associated with on NetSuite.

Data Type

string

Default Value

""

Remarks

Together with User and Password, this field is used to authenticate to NetSuite.

Aggregate Column Mode

Indicating how aggregate columns should be treated.

Data Type

string

Default Value

"Ignore"

Remarks

Aggregate columns are the columns that will appear on base tables which aggregate all of the data contained within child collections. Because these columns include all the data of a child collection, they can become very large. In some situations, such as writing the data to an offline database, it may be advisable to set AggregateColumnMode to either Ignore or List. The data in child tables can still be retrieved by setting IncludeChildTables to true. Setting AggregateColumnMode to List will still cause aggregate columns to be listed for use with inserts and updates.

Ignore	All aggregate will be ignored and will not show up as available colums in the table definition.
List	Aggregate columns will be listed in all tables, but on base tables such as SalesOrders, they will not retrieve data from NetSuite.
ListAndRetrieve	Aggregate columns will be listed and requested on all tables.

Application Id

As of version 2015.2, requests to NetSuite require an application ID or token-based authentication details.

Data Type

string

Default Value

""

Remarks

This field is used to authenticate to NetSuite. These IDs are different from application IDs created in versions 2015.1 and older. You can find your applications in the NetSuite UI under Setup > Integration > Manage Integrations. An application ID is set by default, but can optionally be specified.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description

TUNNEL	80	When this is set, the adapter opens a connection to NetSuite and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Include Child Tables

A boolean indicating if child tables should be displayed.

Data Type

bool

Default Value

false

Remarks

If this is set to true, tables will be displayed for all child lists of a given entity. For instance, the CashRefund table in the NetSuite development environment has a child list called ItemList. Therefore, a new table called CashRefund_ItemList will be displayed if this property is set to true. This can be useful for listing each item in the list in its own row.

Include Custom Field Columns

A boolean indicating if you would like to include custom field columns.

Data Type

bool

Default Value

true

Remarks

Setting this to true will cause custom fields to be displayed directly on tables as their own rows. However, it will cause lower performance when retrieving the table metadata information for the first time on an open connection. Table metadata is stored on the connection and cleared when the connection is closed.

Include Custom List Tables

A boolean indicating if you would like to use tables based on custom lists.

Data Type

bool

Default Value

false

Remarks

Setting this to true will cause custom lists types to be included as their own tables. However, it will cause lower performance when retrieving the table metadata information for the first time on an open connection. Table metadata is stored on the connection and cleared when the connection is closed.

Include Custom Record Tables

A boolean indicating if you would like to use tables based on custom record types.

Data Type

bool

Default Value

true

Remarks

Setting this to true will cause custom record types to be included as their own tables. However, it will cause lower performance when retrieving the table metadata information for the first time on an open connection. Table metadata is stored on the connection and cleared when the connection is closed.

Include Reference Columns

A comma separated list representing the columns to include when retrieving data from a field representing a record reference.

Data Type

string

Default Value

"InternalId, Name"

Remarks

Many fields in NetSuite are references to other types of records. For instance, an Invoice might reference an Account and a Customer record. There are several pieces of data that can be returned by NetSuite when retrieving data from a record reference field. The available values are:

InternalId	The NetSuite foreign key for the record reference.
Name	A readable name for the record referenced.
Type	The type of record referenced. This is not always given a value as the given field may only have one type.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Logout Unknown Sessions

Boolean indicating if unknown sessions already established to the user account should be removed.

Data Type

bool

Default Value

false

Remarks

Most users in NetSuite will get one session they can use to connect to NetSuite Web Services at a time. If one attempts to establish a second session when an existing session is already established, NetSuite will throw an exception. While the NetSuite Adapter attempts to share sessions across threads for the same user, it is still possible to have an established session that prevents the adapter from establishing a new one. For instance, the adapter may have been forcibly terminated, resulting in a session that was never properly closed.

Setting this property to true will cause the adapter to check for existing sessions when it attempts to create a new one. If an existing session is unrecognized by the adapter, it will close it in order to establish its own session. It is recommended to only set this property to true if the adapter is the only tool being used to connect to your user account via the web services.

Maximum Concurrent Sessions

The maximum number of concurrent sessions available for use by the username specified in the connection.

Data Type

string

Default Value

"0"

Remarks

Except under special circumstances, most NetSuite user accounts will be limited to one session at a time. This presents potential issues if executing requests across multiple threads as NetSuite will throw an exception if it detects that another session for the user is attempting to be created without closing the previous session. In most cases the existing session will be shared across multiple threads for the same user.

If the user has a SuiteCloud Plus license from NetSuite, then they do have access to concurrent Web services sessions. You can set the maximum number of concurrent sessions they have access to here. That number should be obtained from NetSuite.

Netsuite Metadata Folder

A path to a directory to download metadata files from NetSuite. Set this for best performance.

Data Type

string

Default Value

""

Remarks

In order to avoid needless overhead, the adapter downloads a number of metadata files from the NetSuite API. If NetsuiteMetadataFolder is not specified then the Location will be used. Always set a value for NetsuiteMetadata for best performance.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Access Token Secret

The OAuth access token secret for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessTokenSecret property is used to connect and authenticate using OAuth. The OAuthAccessTokenSecret is retrieved from the OAuth server as part of the authentication process. It is used with the OAuthAccessToken and can be used for multiple requests until it times out.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

The number of results to return per page from NetSuite.

Data Type

string

Default Value

""

Remarks

The pagesize can control the number of results requested from NetSuite on a given query. Setting a higher pagesize will cause more data to come back in a given request, but may take longer to execute. Setting a smaller pagesize is generally recommended to ensure timeout exceptions do not occur.

Password

The password of the NetSuite user used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with User and AccountId, this field is used to authenticate to NetSuite.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to NetSuite from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Request Memorized Transactions

A boolean indicating if you would like to request memorized transactions when retrieving transactions from NetSuite.

Data Type

bool

Default Value

false

Remarks

Memorized transactions in NetSuite are transactions that have been memorized for potentially being used again at a later time. They can be set up to recur on a regular basis or as a reminder to the user.

Role Id

The RoleId is the InternalId of the role that will be used to log in to NetSuite. Leave empty to use the user's default role.

Data Type

string

Default Value

""

Remarks

Together with User, Password, and AccountId this field is used to authenticate to NetSuite. If no RoleId is specified, the user's default role will be used. Otherwise, a RoleId may be obtained by logging into NetSuite, and navigating to the Setup -> Integration -> Web Services Preferences page. Here you can select a different Web Services Default Role as well as obtain the Id, displayed next to the specified role.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----

The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"300"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition. If Timeout expires and the operation is not yet complete, the adapter throws an exception.

In NetSuite, operations can take a very long time to return if retrieving data from child tables or retrieving data from from a given table with AggregateColumnMode set to ListAndRetrieve. For instance, it is not unheard of for it to take in excess of 10 minutes to retrieve 1000 SalesOrders in a single request from NetSuite when AggregateColumnMode is set to ListAndRetrieve. Unfortunately, this is a limitation of the NetSuite Web Services with no known solution. If you need to work with aggregate columns or child tables, set Timeout to 0, set a small Pagesize, and select specific columns instead of everything.

Use Async Services

A boolean indicating if you would like to use asynchronous services when inserting, updating, and deleting.

Data Type

bool

Default Value

false

Remarks

NetSuite responses can be fairly slow, especially when inserting, updating, or deleting many records at a time. In these situations it is best to use asynchronous services to submit the data. Asynchronous services allow NetSuite to process the data while your application continues executing. The downside of using asynchronous services is that you will need to check the Job in NetSuite to see if NetSuite has finished processing the request and see if there were any issues.

User

The user of the NetSuite account used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with AccountId and Password, this field is used to authenticate to NetSuite.

User Timezone Offset

Your user timezone offset as defined in your NetSuite preferences under Home --> Preferences --> Time Zone. Ex: -8:00.

Data Type

string

Default Value

""

Remarks

NetSuite returns dates offset based on your user timezone preferences. This applies only to Dates, and not to Datetimes, which always return the same way no matter your preferences. We attempt a best effort to automatically correct for this and return dates as they would appear in the NetSuite UI. However, this is not 100% perfect due to how a few time zones are offset by 24 hours or greater depending on daylight savings time. If your user preferences are set to +13:00, +12:00, or -12:00, this value must be specified to get dates to show up correctly. For other offsets it is not required but recommended for best accuracy.

Use Sandbox

A boolean indicating if you would like to use the sandbox instance of your NetSuite account.

Data Type

bool

Default Value

false

Remarks

Set UseSandbox to true to use your sandbox instead of your live data.

Use Sessions

Establish sessions with NetSuite, or instead submit user credentials on each request.

Data Type

bool

Default Value

true

Remarks

If UseSessions is set to true, then user credentials will not be submitted on each request to NetSuite. Instead, a single Login will be attempted to establish the session, and from that point on, the session will be used on each subsequent request. This is a more secure method of communication.

In general, most NetSuite users can only establish a single session with a company file. The NetSuite Adapter is aware of this limitation and attempts to share the session for a given user across connections if multiple threads are active. However, if the same user is being used to connect from multiple machines or multiple different products, then there are likely to be connection problems. If your use case involves using the same user in these situations, it is better to set UseSessions to false and instead submit credentials on each request to avoid leaving sessions open beyond a single request at a time.

Use Upserts

A boolean indicating if you would like to perform an upsert when an insert operation is used.

Data Type

bool

Default Value

false

Remarks

Upserts can be used to potentially update an existing record when inserting. NetSuite handles this by using the ExternalId on a given record. If you perform an insert when the ExternalId you specify exists in NetSuite, it will instead update the corresponding record. Otherwise an insert will take place. Set this value to false to always insert new records regardless.

Version

The version of the NetSuite API in usage. Defaults to 2017_1.

Data Type

string

Default Value

"2017_1"

Remarks

In most cases the version should not need to be changed. However, if you modify the version, please make sure beforehand that the version specified exists as part of the NetSuite API. If an incorrect version is specified, the adapter will be unable to retrieve or update tables in NetSuite.

Web Service Host

An optional override for the web service host such as webservices.na1.netsuite.com.

Data Type

string

Default Value

""

Remarks

This is an override for the web services host if for some reason the host cannot be dynamically resolved. It intended as a last resort and should not normally need to be specified.

In most cases the web service host will be dynamically determined by using the NetSuite web service request that retrieves the correct web service host for your Account as well as the UseSandbox connection property. This will result in one extra request each time you create and establish a new connection. If you would like to avoid this extra request, you can instead set the WebServiceHost to override it. Note that specifying WebServiceHost will cause UseSandbox to be ignored.

The currently accepted web service hosts for NetSuite are the following: webservices.netsuite.com webservices.na1.netsuite.com webservices.sandbox.netsuite.com

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the NetSuite Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to NetSuite

Connecting to NetSuite

Connecting to NetSuite is fairly straightforward. Simply provide the User, Password, and AccountId properties to connect. It is also recommended but not required to specify the RoleId and NetsuiteMetadataFolder: if RoleId is empty, the user's default role is used. The Netsuite Metadata Folder is a folder on disk where NetSuite metadata files will be stored. If you do not specify one, a folder location will be selected automatically. To ensure fast load times when listing metadata about tables, it is best to set this property.

NetSuite Web Services Permissions

The adapter communicates with NetSuite through the NetSuite Web services. This means that the user specified in the connection must have permissions on the specified AccountId to connect through NetSuite Web services. If the user does not already have Web services permissions, an exception stating "You do not have permission to access Web services features" will be thrown when trying to connect. If this happens, an administrator will need to grant Web services permissions to the user by doing the following:

1. Create a Web services role

Log into NetSuite and under Setup go to User/Roles -> Manage Roles -> New.

Click Permissions -> Setup and add the "Web Services" role.

Add other permissions that are needed for interacting with various entities and transactions.

Under Setup, go to User/Roles -> Manage Users and select the user.

On the Access tab, add the newly created role and save the user.

2. Set the user's role to be the WebService default role (optional - can be provided in the connection instead):

Log into NetSuite and under Setup go to Integration -> Web Service Preferences.

Select the user in the Name menu. In the Web Services Default Role menu, select the newly created role.

Click add and save to save changes.

User Sessions

By default, NetSuite allows users to establish only one session when connecting to NetSuite at a time. If a second session is started, then the first session will be invalidated. For this reason, we default UseSessions to false. This will cause each request to be its own session. If you set UseSessions to true, be careful to close each connection before starting a new one. Otherwise you may receive an exception that your session has timed out. Please note that some UI editors take up multiple connections, and setting UseSessions to true will cause this exception in this scenario.

Token Authentication

Instead of using user/password credentials to connect to NetSuite, token authentication can instead be used. This is available as an alternative to user/password due to the expiration that can be set on passwords. Tokens do not expire unless manually reset or removed by an admin. In order to perform Token authentication, do not set the User and Password. Instead, follow these steps to obtain a token and set the following values:

In NetSuite, log in as an administrator role and navigate to Setup --> Company --> Enable Features --> SuiteCloud --> Manage Authentication. Make sure Token-Based Authentication is checked and save changes.

Navigate to Setup --> Integration --> Manage Integrations.

Create a new integration and select Token-Based Authentication.

When the integration is created, the Consumer Key and Consumer Secret displayed will map directly to the

OAuthClientId and OAuthClientSecret connection properties. Write these down.

Create a token role by navigating to Setup --> User/Roles --> Manage Roles and either create a new role or edit an existing role.

Under Permissions --> Setup, the role must have the User Access Token: Full, Access Token Management: Full, and Web Servies: Full permissions.

Add the role to a user under Lists --> Employees --> Employees. Select to edit an employee and add the new token role under Access --> Roles.

Navigate to Setup --> User/Roles --> Access Tokens and create a new access token. Select the application name as the integration that was created earlier, and the same user and role that were updated in the previous steps.

After creating the access token, a Token Id and Token Secret will be displayed. These map directly to the

OAuthAccessToken and OAuthAccessTokenSecret. Write these down.

After creating the access token, a connection can now be made using the values obtained from the previous steps. Specificy these connection properties at a minimum to connect:

AccountId specifying the account to connect to.

OAuthClientId the Consumer Key displayed when the application was created.

OAuthClientSecret the Consumer Secret displayed when the application was created.

OAuthAccessToken the Token Id when the access token was created.

OAuthAccessTokenSecret the Token Secret when the access token was created.

Concurrent Requests

By default, NetSuite allows only one requst at a time for a given user. If a request is still executing against NetSuite when another one is attempted, an error stating "Only one request may be made against a session at a time" will be thrown on the second connection.

NetSuite Response Times

NetSuite response times can take a long time depending on what you are retrieving. This can range from 20-40 seconds to retrieve 1000 SalesOrders to up to 8-12 minutes to retrieve a single page of the SalesOrder_ItemList. The main factor that can affect performance is whether or not child table information is retrieved. For instance, the SalesOrder_ItemList table would be considered the child table of SalesOrder. In addition, the information in the ItemListAggregate on the SalesOrder table would be considered child table information. Because of the significant delay it adds in response times from NetSuite, we have defaulted AggregateColumnMode to Ignore. When set to either Ignore or List, this will cause any Aggregate columns on base tables such as SalesOrder to not request data from NetSuite. This property has no effect on child tables.

Asynchronous Services

Slow NetSuite response times extend to inserts, updates, and deletes as well.This can be especially noticeable when using batch processing. When inserting, updating, or deleting multiple records at a time, it may be worthwhile to set UseAsyncServices to true. This will cause the request to be processed asynchronously on NetSuite's end and a JobId will be returned in the Info#TEMP table. The JobId can be checked against the stored procedures CheckJobStatus and GetJobResults for information about when the job is completed, if any errors occurred, and for the InternalIds for newly created entities.

Exposed Data

The NetSuite Adapter dynamically obtains information about your NetSuite tables at runtime. The tables and columns that are exposed may be controlled by several connection properties. This section will discuss what connection properties may be used to control the listed tables and columns, and how to get the NetSuite data you are looking for. A further discussion about types of tables with examples of how to perform inserts and updates to each type can be found in Tables.

NetSuite Default Record Types

There are a number of default record types that come with NetSuite without any customization. These include entities such as Accounts, Customers, and Vendors. It also includes transactions such as SalesOrders, Invoices, and PurchaseOrders. Each of these default record types will be determined at runtime based on the Version connection property. A table will be exposed for every record type found this way.

Note that these default record tables are also frequently called parent tables when talking about child tables below.

Version

At least twice a year, NetSuite will release a new version. The new version often contains new default record types available to all users. In order to get the latest record types to be exposed as tables, set the Version connection property. The default is updated each major release to be in line with the latest from NetSuite.

Child Tables

Many of the default record types in NetSuite will contain collections of data. For instance, Customers can have several addresses. An Invoice can have several line items. These child collections are exposed in what are called Child Tables. For instance, Invoice_itemList would be considered the child table of the parent table Invoice.

Due to the number of child collections, turning them on will cause the total displayed table count to balloon significantly by several hundred. Child tables can be enabled by setting the IncludeChildTables connection property to true. It is false by default.

Child tables are not exposed beyond the first level. For instance, the Expense List on a Purchase Order containsa Linked Order List. There will not be a table exposed for the Linked Order List. Instead, an aggregate column would need to be used to retrieve this data as explained below.

Aggregate Columns

Child collections may also be exposed in a different way via the AggregateColumnMode connection property. Aggregate columns return collections of complex data such as a collection of addresses or line items. These values would not fit into a single column on their own and are thus aggregated together as XML values. Since most child collection information can be obtained easily and more readably by setting IncludeChildTables to true, aggregates are disabled by default.

Aggregates can be useful for exposing further collections on child tables. For instance, if you wanted to retrieve the Linked Order List from the Expense List on a Purchase Order. Set both IncludeChildTables to true and AggregateColumnMode to List. Those settings will cause the ExpenseList_linkedOrderList column to be displayed and return with data on the PurchaseOrder_expenseList table.

It is not recommended to set AggregateColumnMode to ListAndRetrieve except when trying very small samples, such as if you intend to perform inserts via an aggregate column and need to get a few samples of what the data looks like. Setting AggregateColumnMode to ListAndRetrieve will cause aggregate column data to be requested at all times on all tables. This will significantly slow down performance on parent tables such as Customers, PurchaseOrders, and Invoices due to NetSuite returning data much slower when this data is requested. When AggregateColumnMode is set to List, aggregate columns will always come back with data when retrieving data from child tables since NetSuite will already come back with the data at no additional performance penalty.

Custom Records

Custom records are exposed as full tables. Note that some types of records that are exposed in NetSuite by default are still considered custom records. We consider anything a custom record if it is available in NetSuite under Customization --> Lists, Records, & Fields --> Record Types. To control this setting, change the IncludeCustomRecordTables.

Custom Lists

Custom lists are not exposed by default. Custom lists in NetSuite are lists of information that may be selected for some types of custom fields. These do not typically contain data that is important since it rarely changes. However, these lists can be exposed as tables if needed by setting the IncludeCustomListTables connection property to true.

Custom Fields

Custom fields are exposed by default and will be displayed directly on the appropriate table. Custom fields are displayed for both parent tables such as Invoice and Customer, and also for child tables such as Invoice_itemList and Customer_addressbookList. Custom Field exposure may be controlled with the IncludeCustomFieldColumns connection property.

Reference Columns

Reference columns are columns that include a record reference. For instance, Invoice contains an Entity record reference. Record references always come back with multiple pieces of information such as an internalid, name, and potentially type and externalid. Due to the number of record reference columns in NetSuite, by default they only display with InternalId and Name. This can be modified by changing the IncludeReferenceColumns connection property.

Reports

Reports are not supported in the NetSuite API. This is a limitation of the API which means that reports are not available for us or other third party tools. NetSuite instead recommends customers make use of the PostingTransactionSummary for getting the sort of information that would appear on reports. The PostingTransactionSummary view can be used to access that information.

Saved Searches

NetSuite saved searches are supported, although there are some limitations due to the NetSuite api. Please see Saved Searches for more information.

Saved Searches

Saved searches are supported with some limitations due to NetSuite restrictions.

SavedSearches View

The SavedSearches view is a good way to retrieve a list of your available saved searches for a given search type. A SearchType must be specified to retrieve information from the SavedSearches view. For instance:

```
SELECT * FROM SavedSearches WHERE SearchType='Transaction'
```

CreateSavedSearchSchema Stored Procedure

The CreateSavedSearchSchema stored procedure is used to generate a schema file for the saved search. This schema file is automatically generated based on the saved search results. The file will be written to the folder indicated by the Location connection property.

To execute the stored procedure, simply supply the Name, SearchType, and Id (returned from SavedSearches):

```
EXEC CreateSavedSearchSchema @SavedSearch='MySavedSearch',
@SearchType='Transaction', @SearchId='12345'
```

Unlike normal tables, there is no metadata request for saved searches. To get the metadata, we have to retrieve a sample of the results and make an assumption about the available columns based on the response. Sometimes this may mean columns are missing due to all of their values being null in the sample response.

It is easy to modify these schema files. Just open them with any text editor. You will notice the file is just XML, with the column names defined near the top of the file. You can rename the columns to something more appropriate. Just be aware that valid column names for these XML files must be alphanumeric, must start with an alphabetical character, and may only have the underscore '_' character in addition to alphanumeric characters.

Retrieving Data

Once a schema file has been created, data can be retrieved from the saved search using the schema name. For instance:

```
SELECT * FROM MySavedSearch
```

You may notice that the column names do not match what you see displayed in the UI. This is because NetSuite is using labels when you execute the saved search. Sometimes those columns come from different tables (when there is a join), and sometimes they are custom fields. Since there is no metadata service available for saved searches, there is not a good way to automatically obtain these labels. However, as explained earlier, these schema files may by updated manually to express the column names the way you want them to.

Limitations

NetSuite imposes a few limitations on saved searches. Calcualted columns (formulas) cannot be retrieved via the NetSuite API. Only columns directly from the table may be retrieved. If you have a calculation in your saved search, you would need to return each individual column used in the calculation and perform the calculation client side. Ie:

```
SELECT (col1 / col2) AS calc FROM MySavedSearch
```

Saved searches that return an aggregation or summary (Group / Count / Sum / Minimum / Maximum / Average) cannot be returned at all. The NetSuite API will throw an exception upon attempting to retrieve these saved searches. Due to this limitation, these saved searches are unavailable for our and any other third party tool.

SQL Compliance

The NetSuite Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the NetSuite API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key InternalId is required to update a record. See UPDATE Statements for a syntax reference and examples.

UPSERT Statements

An UPSERT updates a record if it exists and inserts the record if it does not. See UPSERT Statements for a syntax reference and examples.

DELETE Statements

The primary key InternalId is required to delete a record. See DELETE Statements for a syntax reference and examples.

GETDELETED Statements

GETDELETED statements return the Ids of deleted records. See GETDELETED Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the NetSuite adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  } [ , ... ]
  [
    JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | < | > | >= | <= | LIKE | IN | NOT IN
| AND | OR | IS NULL | IS NOT NULL } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Account
```

Rename a column:
```
SELECT "AcctName" AS MY_AcctName FROM Account
```

Search data:
```
SELECT * FROM Account WHERE acctName = 'Checking';
```

The NetSuite APIs support the following operators in the WHERE clause: =, !=, <, >, >=, <=, LIKE, IN, NOT IN, AND, OR, IS NULL, IS NOT NULL.
```
SELECT * FROM Account WHERE acctName = 'Checking';
```

Return the number of items matching the query criteria:
```
SELECT COUNT(*) AS MyCount FROM Account
```

Retrieve data from multiple tables.
```
SELECT n.Note AS MyNote, c.InternalId AS CustomerId FROM Customer,
Note
```

See JOIN Queries for details.

JOIN Queries

This section discusses some of the features and restrictions that are specific to how the adapter supports joins. If possible the NetSuite Adapter will attempt to perform joins directly in NetSuite. If NetSuite cannot handle the join, it will instead be performed client side when SupportEnhancedSQL is set to true.

The NetSuite supports join queries on tables that have a relationship defined in NetSuite. Only left outer joins are supported directly by NetSuite.

When you use the adapter to join tables with SQL, this is the same as creating a Saved Search in the NetSuite UI and then selecting the option to get back data from other related tables.

The following query retrieves all Notes associated with each Customer:
```
SELECT n.Note AS MyNote, c.InternalId AS CustomerId
FROM Customer c
LEFT OUTER JOIN Note n
ON n.Entity_InternalId = c.InternalId
```

Joins of more than two tables are supported. The following query retrieves Customer and BillingSchedule information about each SalesOrder:
```
SELECT Customer.AccountNumber AS CustomerAcctNumber,
BillingSchedule.InitialAmount AS InitialAmount,
SalesOrder.InternalId AS SalesOrderId
FROM SalesOrder
LEFT OUTER JOIN Customer ON SalesOrder.Entity_InternalId =
Customer.InternalId
```

```
LEFT OUTER JOIN BillingSchedule ON
SalesOrder.BillingSchedule_InternalId=BillingSchedule.InternalId
```

Due to limitations in the NetSuite API, joins are not supported for the following tables:

Child tables

Custom record tables

Tables listed in the "Simple Tables" section in

Tables

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT InternalId, AcctName INTO "csv://Account.txt" FROM "Account"
WHERE acctName = 'Checking'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT InternalId, AcctName INTO "csv://Account.txt;delimiter=tab"
FROM "Account" WHERE acctName = 'Checking'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )
```

```
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the InternalId of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Account (AcctName) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "Petty Cash");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("InternalId"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { InternalId = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Account SET AcctName='Petty Cash' WHERE
InternalId = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { InternalId
= <expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:netsuite:Account
Id=XABC123456;Password=password;User=user;Role
Id=3;Version=2013_1;Location=C:\\myfolder\\;",);
String cmd = "DELETE FROM Account WHERE InternalId = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "");
int count=pstmt.executeUpdate();
connection.close();
```

GETDELETED Statements

You can issue the GETDELETED query to retrieve all records deleted from the live data for the time range specified. This query accepts a datetime value as a filter.

```
GETDELETED FROM <table_name> WHERE <search_condition>

<search_condition> ::=
  {
    <expression> { = | < | <= | > | >= } [ <expression> ]
  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

```
String cmd = "GETDELETED FROM Account WHERE TimeModified
>='2013-01-01' AND TimeModified <='2013-02-01'";
PreparedStatement pstmt = connection.prepareStatement(cmd);
ResultSet rs = pstmt.getResultSet();
while(rs.next()){
  System.out.println(rs.getString("InternalId"));
}
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV OData Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to OData

To connect, you need to set the Url to a valid OData service root URI in addition to the authentication values.

Fine-Tuning Data Access

Set the following properties to control how the adapter models OData APIs as a database:

NavigationPropertiesAsViews: By default, the adapter models navigation properties as views. This enables access to related entities, even though these entities may not be linked by a foreign key in your OData service.

SupportsExpand: If your API does not support the *$expand* parameter, set this property to avoid an error when NavigationPropertiesAsViews is set. If this is the case for your API, specify the base entity's primary key in the WHERE clause to access navigation properties.

DataFormat: Set this property to JSON or XML. Otherwise, the adapter uses the default format defined by the service.

ODataVersion: Use this to override the version detected by the adapter. This is useful if your application supports an older OData version.

UseIdUrl: By default the adapter returns the direct URL to an entity as the primary key. By setting this to false, the entity key is returned.

UseSimpleNames: Set this to true to return only alphanumeric characters in column names. This can help you to avoid SQL escapes and errors in SQL-based tools.

Authenticating to OData

The adapter supports the major authentication schemes, including HTTP and Windows.

Set AuthScheme to use the following authentication types.

The adapter simplifies OAuth configuration. See Using OAuth for a how-to.

HTTP Authentication

The adapter supports authentication with HTTP Basic, Digest, and custom headers. To use Basic or Digest, set the User and Password. You can specify other authentication values in CustomHeaders.

Windows (NTLM)

Set the Windows User and Password to connect and set AuthScheme to "NTLM".

Kerberos and Kerberos Delegation

To authenticate with Kerberos, set AuthScheme to NEGOTIATE. To use Kerberos Delegation, set AuthScheme to KERBEROSDELEGATION. If needed, provide the User, Password, and KerberosSPN. By default, the adapter attempts to communicate with the SPN at the specified Url.

Securing OData Connections

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Allow Resource Url	A boolean indicating if the direct URL to an entity set may be used instead of the service document.
Auth Scheme	The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, NEGOTIATE, or SHAREPOINTONLINE.
Azure AD Resource	The Azure Active Directory resource to authenticate to (only used with Azure AD OAuth).
Azure AD Tenant	The Azure Active Directory tenant to authenticate against (only used with Azure AD OAuth).
Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Continue On Error	Whether or not to continue after encountering an error on a batch request.
Cookies	Allows cookies to be manually specified in name=value pairs separated by a semicolon.
Custom Headers	Other headers as determined by the user (optional).
Custom Url Params	The custom query string to be included in the request.
Data Format	The data format to retrieve data in. Select either ATOM or JSON.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Kerberos SPN	The service principal name for the Kerberos domain controller.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

Navigation Properties As Views	A boolean indicating navigation properties should be promoted to full views.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Access Token Secret	The OAuth access token secret for connecting using OAuth.
OAuth Access Token URL	The URL to retrieve the OAuth access token from.
OAuth Authorization URL	The authorization URL for the OAuth service.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Grant Type	The grant type for the OAuth flow. Can be either CLIENT, CODE or PASSWORD.
OAuth Params	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Refresh Token URL	The URL to refresh the OAuth token from.
OAuth Request Token URL	The URL the service provides to retrieve request tokens from. Required in OAuth 1.0.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
OAuth Version	The version of OAuth being used. Enter 1.0 or 2.0.
OData Version	The version of OData to use. By default the provider will attempt to autodetect the version.

Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password used to authenticate to the OData site.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to OData from the provider.
SAP Company DB	Your SAP Business One company database. Only used when AuthScheme is set to SAPBusinessOne.
SSL Client Cert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSL Client Cert Password	The password for the TLS/SSL client certificate.
SSL Client Cert Subject	The subject of the TLS/SSL client certificate.
SSL Client Cert Type	The type of key store containing the TLS/SSL client certificate.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.

Supports Expand	Whether you need to specify the base entity's key to query navigation property views.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Url	URL to the Organization root or the OData services file. For example, http://MySite/MyOrganization.
Use Client Side Paging	Whether or not the CData ADO.NET Provider for OData should use client side paging.
Use Etags	Whether or not the OData source uses Etags.
Use Id Url	Boolean determining if the Id column representing the direct URL to a given entity will be displayed.
User	The user who is authenticating to the OData site.
Use Simple Names	Whether or not to use simple names for tables and columns.

Allow Resource Url

A boolean indicating if the direct URL to an entity set may be used instead of the service document.

Data Type

bool

Default Value

false

Remarks

A properly implemented OData service will list all entity sets in the service document. However, since anyone can make an OData service, some services are not correctly implemented and do not list all entity sets available in the service document. If this is the case for your service, set this property to true and specify the Url as the full url to the specific entity set you would like to work with.

Auth Scheme

The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, NEGOTIATE, or SHAREPOINTONLINE.

Data Type

string

Default Value

"NONE"

Remarks

Together with Password and User, this field is used to authenticate against the OData server. NONE is the default option.

NTLM: Set this to use your Windows credentials for authentication.

BASIC: Set this to use HTTP Basic authentication.

DIGEST: Set this to use HTTP Digest authentication.

NEGOTIATE: If

AuthScheme is set to NEGOTIATE, the adapter will negotiate an authentication mechanism with the server. Set AuthScheme to NEGOTIATE if you want to use Kerberos authentication.

KERBEROSDELEGATION: Set this to use delegation through the Kerberos protocol. Set the

User and Password of the account you want to impersonate.

SAPBUSINESSONE: Set this to use SAP Business One authentication.

SHAREPOINTONLINE: Set this to use SharePoint Online authentication.

Azure AD Resource

The Azure Active Directory resource to authenticate to (only used with Azure AD OAuth).

Data Type

string

Default Value

""

Remarks

The resource must be specified if using Azure Active Directory OAuth. It should be set to the App ID URI of the web API (secured resource).

Azure AD Tenant

The Azure Active Directory tenant to authenticate against (only used with Azure AD OAuth).

Data Type

string

Default Value

"common"

Remarks

The tenant must be specified if using Azure Active Directory OAuth. The tenant is used to control who can sign into the application. This should be the name of the tenant such as xxx.onmicrosoft.com, the id such as 8eaef023-2b34-4da1-9baa-8bc8c9d6a490, contoso.onmicrosoft.com, or the word common.

Callback URL

The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.

Data Type

string

Default Value

""

Remarks

During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

Continue On Error

Whether or not to continue after encountering an error on a batch request.

Data Type

bool

Default Value

false

Remarks

This connection property is only supported on servers with OData version 4.0 and higher. However, individual servers may choose to ignore this setting. Setting ContinueOnError to true will cause exceptions to be returned in the temporary table instead of being thrown when a batch request is attempted.

Cookies

Allows cookies to be manually specified in name=value pairs separated by a semicolon.

Data Type

string

Default Value

""

Remarks

In general it should not be required to set this property. However, there are many different flavors of OData services. If your solution requires cookies that are obtained outside of the OData Adapter, they can be manually specified here. Specify cookies in name=value pairs separated by a semicolon. For instance: Cookie1=value;Cookie2=value2.

Custom Headers

Other headers as determined by the user (optional).

Data Type

string

Default Value

""

Remarks

This property can be set to a string of headers to be appended to the HTTP request headers created from other properties, like ContentType, From, etc.

The headers must be of the format "header: value" as described in the HTTP specifications. Header lines should be separated by CRLF (the carriage return and line feed characters).

Use this property with caution. If this property contains invalid headers, HTTP requests may fail.

This property is useful for fine-tuning the functionality of the adapter to integrate with specialized or nonstandard APIs.

This property can be set to a string of headers to be appended to the HTTP request headers created from other properties like ContentType, From, etc.

The headers must be of the format "header: value" as described in the HTTP specifications. Header lines should be separated by CRLF (the carriage return and line feed characters).

Use this property with caution. If this property contains invalid headers, HTTP requests may fail.

This property is useful for fine-tuning the functionality of the adapter to integrate with specialized or nonstandard APIs.

Custom Url Params

The custom query string to be included in the request.

Data Type

string

Default Value

""

Remarks

The CustomUrlParams allow you to specify custom query string parameters that are included with the HTTP request. The parameters must be encoded as a query string in the form field1=value1&field2=value2&field3=value3, etc. The values in the query string must be URL encoded.

Data Format

The data format to retrieve data in. Select either ATOM or JSON.

Data Type

string

Default Value

""

Remarks

Note that not all data sources support JSON. Other IANA content types are not supported at this time. Leave blank to use the system service default. If blank, ATOM will be used when submitting data in an insert or update.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to OData and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.

SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Kerberos SPN

The service principal name for the Kerberos domain controller.

Data Type

string

Default Value

""

Remarks

If the service principal name on the Kerberos domain controller is not the same as the URL that you are authenticating to, the service principal name should be set here.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Navigation Properties As Views

A boolean indicating navigation properties should be promoted to full views.

Data Type

bool

Default Value

true

Remarks

This property can be useful for OData services that can return related collections of entities, or navigation properties. Some OData entities can only be accessed through navigation properties. NavigationPropertiesAsViews will cause all of the discovered navigation properties to be added as views in the format *ParentTable_NavigationProperty*.

Retrieving Data from Limited OData APIs

In most cases, NavigationPropertiesAsViews can be left on and the resulting views can be accessed with any SELECT query. However, some OData APIs have limitations that require you to specify the primary key of the parent record when querying a navigation property.

For example:

```
SELECT * FROM Categories_Products WHERE Categories_CategoryId='1'
```

You will also need to set SupportsExpand to false. You can find more information on this API limitation in the documentation for the property.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Access Token Secret

The OAuth access token secret for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessTokenSecret property is used to connect and authenticate using OAuth. The OAuthAccessTokenSecret is retrieved from the OAuth server as part of the authentication process. It is used with the OAuthAccessToken and can be used for multiple requests until it times out.

OAuth Access Token URL

The URL to retrieve the OAuth access token from.

Data Type

string

Default Value

""

Remarks

The URL to retrieve the OAuth access token from. In OAuth 1.0 the authorized request token is exchanged for the access token at this URL.

OAuth Authorization URL

The authorization URL for the OAuth service.

Data Type

string

Default Value

""

Remarks

The authorization URL for the OAuth service. At this URL the user logs into the server and grants permissions to the application. In OAuth 1.0 if permissions are granted the request token is authorized.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Grant Type

The grant type for the OAuth flow. Can be either CLIENT, CODE or PASSWORD.

Data Type

string

Default Value

"CODE"

Remarks

The grant type for the OAuth flow. Can be either CLIENT, CODE or PASSWORD.

OAuth Params

A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.

Data Type

string

Default Value

""

Remarks

A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Refresh Token URL

The URL to refresh the OAuth token from.

Data Type

string

Default Value

""

Remarks

The URL to refresh the OAuth token from. In OAuth 2.0 this URL is where the refresh token is exchanged for a new access token when the old access token expires.

OAuth Request Token URL

The URL the service provides to retrieve request tokens from. Required in OAuth 1.0.

Data Type

string

Default Value

""

Remarks

The URL the service provides to retrieve request tokens from. Required in OAuth 1.0. In OAuth 1.0 this is the URL where the app makes a request for the request token.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\OData Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

OAuth Version

The version of OAuth being used. Enter 1.0 or 2.0.

Data Type

string

Default Value

""

Remarks

The version of OAuth being used. Enter 1.0 or 2.0.

OData Version

The version of OData to use. By default the provider will attempt to autodetect the version.

Data Type

string

Default Value

"AUTO"

Remarks

The version of OData to use. By default the adapter will automatically attempt to determine the version the service is using. If a version cannot be resolved, 3.0 will be used. This can optionally be manually set.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password used to authenticate to the OData site.

Data Type

string

Default Value

""

Remarks

The password used to authenticate to the OData site.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to OData from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SAP Company DB

Your SAP Business One company database. Only used when AuthScheme is set to SAPBusinessOne.

Data Type

string

Default Value

""

Remarks

This property is required when connecting to SAP Business One, but otherwise can be ignored.

SSL Client Cert

The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The SSLClientCertType field specifies the type of the certificate store specified by SSLClientCert. If the store is password protected, specify the password in SSLClientCertPassword.

SSLClientCert is used in conjunction with the SSLClientCertSubject field in order to specify client certificates. If SSLClientCert has a value, and SSLClientCertSubject is set, a search for a certificate is initiated. Please refer to the SSLClientCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.
CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

SSL Client Cert Password

The password for the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

SSL Client Cert Subject

The subject of the TLS/SSL client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning

CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

SSL Client Cert Type

The type of key store containing the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.

PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.
SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Supports Expand

Whether you need to specify the base entity's key to query navigation property views.

Data Type

bool

Default Value

true

Remarks

This connection property is primarily used with limited OData APIs; it determines whether navigation properties can be retrieved from the base entity set. In OData, navigation properties link a base entity to a related entity or a collection of related entitites.

For more on navigation properties, see Data Model.

Working with Limited APIs

In OData, the *$expand* parameter is used to expand specified navigation properties when requesting data from a given entity set. In SQL, this makes it possible to execute a *SELECT* * to a navigation property view.

If *$expand* is not supported, a different request must be made to retrieve a navigation property, one that specifies the primary key of the base entity set. This API restriction is reflected in SQL: You will need to specify the base entity's primary key in the WHERE clause.

For example, consider two entities with a one-to-many relationship in the Northwind sample service, Categories and Products. In OData, the Products associated with a given Category could be represented as a navigation property on the base Category entity set. The adapter models the Products navigation property as a Categories_Products view.

If *$expand* is not supported, use a query like the following to this view:

```
SELECT      *
FROM        Categories_Products
WHERE       (Categories_CategoryID = 1)
```

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Url

URL to the Organization root or the OData services file. For example, http://MySite/MyOrganization.

Data Type

string

Default Value

""

Remarks

URL to the Organization root or the OData services file. For example, http://MySite/MyOrganization.

Use Client Side Paging

Whether or not the CData ADO.NET Provider for OData should use client side paging.

Data Type

bool

Default Value

false

Remarks

Some sources do not support server side paging. In these cases, set UseClientSidePaging to true. Otherwise, leave it as false. Setting UseClientSidePaging to true on a source that already supports paging can cause incomplete results.

Use Etags

Whether or not the OData source uses Etags.

Data Type

bool

Default Value

false

Remarks

Some OData sources do not use Etags. In these instances, set UseEtags to False.

Use Id Url

Boolean determining if the Id column representing the direct URL to a given entity will be displayed.

Data Type

bool

Default Value

false

Remarks

Boolean determining if the Id column representing the direct URL to a given entity will be displayed.

If set to false, the entity key is returned instead. For example, the adapter returns "1" as the primary key for the entity specified with the URL below:
`http://host/service/Categories(1)`

User

The user who is authenticating to the OData site.

Data Type

string

Default Value

""

Remarks

The user who is authenticating to the OData site.

Use Simple Names

Whether or not to use simple names for tables and columns.

Data Type

bool

Default Value

false

Remarks

This connection property controls whether simple names will be used for tables and columns. This means that tables and columns will be output as alphanumeric. Underscores will be used to replace any unconventional characters for a SQL identifier such such as commas, colons, spaces, etc.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the OData Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting Using OAuth

OData uses OAuth for authentication. OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Obtain OAuth URLs

You will need the following URLs to complete the OAuth interaction. These URLs are often obtained from the API reference for your data source.

OAuthRequestTokenURL: Required for OAuth 1.0. In OAuth 1.0 this is the URL where the app makes a request for the request token.

OAuthAuthorizationURL: Required for OAuth 1.0 and 2.0. This is the URL where the user logs into the service and grants permissions to the application. In OAuth 1.0 if permissions are granted the request token is authorized.

OAuthAccessTokenURL: Required for OAuth 1.0 and 2.0. This is the URL where the request for the access token is made. In OAuth 1.0 the authorized request token is exchanged for the access token.

OAuthRefreshTokenURL: Required for OAuth 2.0. In OAuth 2.0 this is the URL where the refresh token is exchanged for a new access token when the old one expires. Note that for your data source this may be the same as the access token URL.

CallbackURL: Required depending on your data source; your data source may require you to define this URL when you create an app. This is the URL you want to be used as a trusted redirect URL (also called a callback URL), where the user will return with the token that verifies that they have granted your app access. Note that your data source may require the port.

Configuring the Adapter

Use the following properties to configure the adapter for OAuth:

OAuthVersion: Set this to 1.0 or 2.0.

AuthScheme: Set this property to NONE when using OAuth.

OAuthGrantType: Set this to one of the following:

CODE	This is the most common grant type and is used with the InitiateOAuth property. Use this grant type when you do not have the username and password or do not wish to require them.
PASSWORD	This grant type involves setting the username and password and sending that in the request to obtain a token (rather than logging into the account directly like the CODE grant type). This method should only be used if the CODE grant type is not available.

InitiateOAuth: Set this property to OFF, REFRESH, or GETANDREFRESH. For more information, see the following sections.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and OData.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from OData and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If OData issues a long-lived access token, use the OData developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

Advanced Settings

Customizing API Requests

The following properties provide the granular control useful for integrating with nonstandard APIs or to access more advanced OData functionality.

CustomUrlParams: Set this to append query string parameters onto the request that the adapter builds.

AllowResourceUrl: Some OData services do not report all available entity sets in the service document, which the adapter uses to model the available entities. If this is the case for your API, set this property and set Url to the full URL of the entity set you want to work with.

Note that if this property is not set you must set Url to the service document to avoid an error.

ContinueOnError: The adapter builds batch requests to OData 4.0 services when batch APIs in the underlying driver interface are invoked; for example, when your application makes a batch request.

When this property is set, errors are returned in a temporary table to avoid breaking execution.

UseEtags: Etags can be used by OData clients to check if a resource has changed on the server and avoid concurrency problems. If you do not need to surface this functionality or if your OData service does not return Etags, set this property to false.

Cookies: If you need to provide cookies obtained outside the adapter, you can set them in this value.

CustomHeaders: You can use this property to add any value to any HTTP request header.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, you do not need to set any additional connection properties. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to network traffic, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain.)

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The OData Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the OData API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the OData adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
    [
      { OFFSET | , }
      <expression>
    ]
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | > | < | >= | <= | IS NULL | IS NOT
NULL | LIKE | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Lead
```

Rename a column:

```
SELECT "FullName" AS MY_FullName FROM Lead
```

Search data:
```
SELECT * FROM Lead WHERE FirstName <> 'Bartholomew';
```

The OData APIs support the following operators in the WHERE clause: =, !=, >, <, >=, <=, IS NULL, IS NOT NULL, LIKE, AND, OR.
```
SELECT * FROM Lead WHERE FirstName <> 'Bartholomew';
```

Return the number of items matching the query criteria:
```
SELECT COUNT(*) AS MyCount FROM Lead
```

Sort a result set in ascending order:
```
SELECT Id, FullName FROM Lead  ORDER BY FullName ASC
```

Predicate Functions

CEILING(value)

Returns the value rounded up to the nearest whole number (no decimal component).

expression: The value to round.

CONCAT(string_expr1, string_expr2)

Returns the string that is the concatenation of string_expr1 and string_expr2.

string_expr1: The first string to be concatenated.

string_expr2: The second string to be concatenated.

CONTAINS(string_expression, string_search)

Returns true if string_expression contains string_expression, otherwise returns false.

string_expression: The string expression to search within.

string_search: The value to search for.

DATE(datetime_offset)

Returns the current date using the specified datetime_offset.

datetime_offset: The datetime offset to use when retrieving the current date.

DAY(datetime_date)

Returns the integer that specifies the day component of the specified date.

datetime_date: The datetime string that specifies the date.

ENDSWITH(string_expression, string_suffix)

Returns true if string_expression ends with string_suffix, otherwise returns false.

string_expression: The string expression to search within.

string_suffix: The string suffix to search for.

FLOOR(value)

Returns the value rounded down to the nearest whole number (no decimal component).

value: The value to round.

FRACTIONALSECONDS(datetime_time)

Returns the decimal value that specifies the fractional seconds component of the specified time.

datetime_time: The datetime string that specifies the time.

HOUR(datetime_time)

Returns the integer that specifies the hour component of the specified time.

datetime_time: The datetime string that specifies the time.

INDEXOF(string_expression, string_search)

Returns the index location where string_search is contained within string_expression.

string_expression: The string expression to search within.

string_search: The search value to locate within string_expression.

ISOF(string_expression, string_type)

Returns true if the string_expression is assignable to type string_type, otherwise returns false.

string_expression: The string expression to check the type of.

string_type: The name of the type.

LENGTH(string_expression)

Returns the number of characters of the specified string expression.

string_expression: The string expression.

MAXDATETIME()

Returns the latest possible datetime.

MINDATETIME()

Returns the earliest possible datetime.

MINUTE(datetime_time)

Returns the integer that specifies the minute component of the specified time.

datetime_time: The datetime string that specifies the time.

MONTH(datetime_date)

Returns the integer that specifies the month component of the specified date.

datetime_date: The datetime string that specifies the date.

NOW()

Returns the current datetime.

REPLACE(string_expression, string_search, string_replace)

Returns the string after replacing any found string_search values with string_replace.

string_expression: The string expression to perform a replace on.

string_search: The string value to find within string_expression.

string_replace: The string value replace and string_search instances found.

ROUND(value)

Returns the value to the nearest whole number (no decimal component).

value: The value to round.

SECOND(datetime_time)

Returns the integer that specifies the second component of the specified time.

datetime_time: The datetime string that specifies the time.

STARTSWITH(string_expression, string_prefix)

Returns true if string_expression starts with string_prefix, otherwise returns false.

string_expression: The string expression to search within.

string_prefix: The string prefix to search for.

SUBSTRING(string_expression, integer_start [,integer_length])

Returns the part of the string with the specified length; starts at the specified index.

expression: The character string.

start: The positive integer that specifies the start index of characters to return.

length: The positive integer that specifies how many characters will be returned.

SUBSTRINGOF(string_expression, string_search)

Returns true if string_expression contains string_expression, otherwise returns false.

string_expression: The string expression to search within.

string_search: The value to search for.

TIME(datetime_offset)

Returns the current time using datetime_offset.

datetime_offset: The datetime offset.

TOLOWER(string_expression)

Returns the string_expression with the uppercase character data converted to lowercase.

string_expression: The string expression to lowercase.

TOTALOFFSETMINUTES(datetime_date)

Returns the integer that specifies the offset minutes component of the specified date.

datetime_date: The datetime string that specifies the date.

TOTALSECONDS(duration)

Returns the duration value in total seconds.

string_duration: The duration.

TOUPPER(string_expression)

Returns the string_expression with the lowercase character data converted to uppercase.

string_expression: The string expression to uppercase.

TRIM(string_expression)

Returns the string_expression with the leading and trailing whitespace removed.

string_expression: The string expression to trim.

YEAR(datetime_date)

Returns the integer that specifies the year component of the specified date.

datetime_date: The datetime string that specifies the date.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, FullName INTO "csv://Lead.txt" FROM "Lead" WHERE
FirstName = 'Bartholomew'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, FullName INTO "csv://Lead.txt;delimiter=tab" FROM "Lead"
WHERE FirstName = 'Bartholomew'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Lead (FullName) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
```

```
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Lead SET FullName='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1045625d-99ee-e011-a272-00155d01ad6b");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]
```

```
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:odata:User=myuseraccount;Passwor
d=mypassword;URL=http://myserver/myOrgRoot;",);
String cmd = "DELETE FROM Lead WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1045625d-99ee-e011-a272-00155d01ad6b");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV RSS Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

The RSS Adapter supports connecting to RSS and Atom feeds, as well as feeds with custom extensions. To connect to a feed, set the URL property. The adapter also supports accessing secure feeds. A variety of authentication mechanisms are supported. See the connection properties for details. The SQLite JDBC driver is used to sort and filter RSS feeds. You will need to include this .jar file in the classpath of your projects. The driver is included in the lib folder in the installation directory.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Token	The token used for authentication.
Firewall Password	A password, if authentication is required to connect through a firewall.
Firewall Port	The TCP port for the firewall FirewallServer -- see the description of the FirewallServer option for details.
Firewall Server	Specify a firewall name or IP address to authenticate requested connections, if necessary.

Firewall Type	The type of firewall to connect through.
Firewall User	The user name to authenticate with the firewall.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password to use during authentication.
Proxy Auth Scheme	The proxy server authorization scheme (default: BASIC).
Proxy Auto Detect	This indicates whether to use the default system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Password	A password, if authenticating with a proxy server.
Proxy Port	The TCP port for the proxy ProxyServer (default: 80).
Proxy Server	If a proxy server is given, then the HTTP request is sent to the proxy instead of the specified server.
Proxy SSL Type	The SSL type to use when connecting to the proxy server (default: AUTO).
Proxy User	A user name, if authentication is to be used for the proxy.
Readonly	You can use this property to enforce read-only access to RSS from the provider.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Support Enhanced SQL	If the property is set to true, the provider will be able to handle SQL queries using a temporary database.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
URL	The URL of the feed.
User	The username to use during authentication.

Auth Token

The token used for authentication.

Data Type

string

Default Value

""

Remarks

An authtoken is a unique secret identifier that can be used to identify the validity of an HTTP request. Several applications and servers use this mechanism to secure feeds. The authtoken set in the connection will be posted with the HTTP server as the request variable '@authtoken'.

Firewall Password

A password, if authentication is required to connect through a firewall.

Data Type

string

Default Value

""

Remarks

If FirewallServer is specified, the FirewallUser and FirewallPassword properties are used to connect and authenticate to the given firewall.

Firewall Port

The TCP port for the firewall FirewallServer -- see the description of the FirewallServer option for details.

Data Type

string

Default Value

""

Remarks

Note that the adapter sets the FirewallPort to the default port associated with the specified FirewallType. See the description of the FirewallType option for details.

Firewall Server

Specify a firewall name or IP address to authenticate requested connections, if necessary.

Data Type

string

Default Value

""

Remarks

If this property is set to a domain name, a DNS request is initiated and the name is translated to the corresponding IP address.

Firewall Type

The type of firewall to connect through.

Data Type

string

Default Value

"NONE"

Remarks

The applicable values are:

Firewall Type	Default FirewallPort
TUNNEL	80
SOCKS4	1080
SOCKS5	1080

Firewall User

The user name to authenticate with the firewall.

Data Type

string

Default Value

""

Remarks

If the FirewallServer is specified, the FirewallUser and FirewallPassword properties are used to connect and authenticate against the firewall.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files used in your application must be deployed with other assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source. Some of the other properties are:

Culture=en-US	Overrides the region settings of the machine to change the default output format of values from the adapter.
DefaultColumnSize=255	Sets the default length of a string field for a provider.

Password

The password to use during authentication.

Data Type

string

Default Value

""

Remarks

Together with User, this field is used to authenticate with the server.

Proxy Auth Scheme

The proxy server authorization scheme (default: BASIC).

Data Type

string

Default Value

""

Remarks

This value may be BASIC, DIGEST, NONE, NTLM, NEGOTIATE or PROPRIETARY.

Proxy Auto Detect

This indicates whether to use the default system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

This indicates whether to use the default system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Proxy Password

A password, if authenticating with a proxy server.

Data Type

string

Default Value

""

Remarks

If the ProxyServer is specified, the ProxyUser and ProxyPassword properties are used to connect and authenticate against the firewall.

Proxy Port

The TCP port for the proxy ProxyServer (default: 80).

Data Type

string

Default Value

""

Remarks

See the description of the ProxyServer field for details.

Proxy Server

If a proxy server is given, then the HTTP request is sent to the proxy instead of the specified server.

Data Type

string

Default Value

""

Remarks

If this property is set to a domain name, a DNS request is initiated and the name is translated to the corresponding address.

Proxy SSL Type

The SSL type to use when connecting to the proxy server (default: AUTO).

Data Type

string

Default Value

""

Remarks

This value may be AUTO, ALWAYS, NEVER, or TUNNEL.

Proxy User

A user name, if authentication is to be used for the proxy.

Data Type

string

Default Value

""

Remarks

If a ProxyServer is specified, the ProxyUser and ProxyPassword options are used to connect and authenticate against the firewall.

Readonly

You can use this property to enforce read-only access to RSS from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected. This can take the form of a full PEM certificate, the path to a file containing the certificate, the public key, the MD5 thumbprint, or the SHA1 thumbprint. If not specified, any trusted certificate will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Support Enhanced SQL

If the property is set to true, the provider will be able to handle SQL queries using a temporary database.

Data Type

bool

Default Value

true

Remarks

If the property is set to true, the adapter will be able to handle SQL queries using a temporary database.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

URL

The URL of the feed.

Data Type

string

Default Value

""

Remarks

If this connection property is not specified, the pseudo column URL in the tables must be specified to a valid feed URL. Stored procedures have a URL parameter that takes precedence over the value of this connection property.

User

The username to use during authentication.

Data Type

string

Default Value

""

Remarks

Together with Password, this field is used to authenticate with the server.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.
`Verbosity=4;`

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the RSS Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the CIS Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\Cisco Systems\CIS 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The CIS Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at: C:\Program Files\Cisco Systems\CIS 7.0\bin. Note reauthenticating to the CIS Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the CIS Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Retrieving Data from Feeds

Connect to Standard RSS or Atom Feeds

To connect to a feed, you can simply set the URL property and obtain data from the RSSFeed or the AtomFeed table. These tables display the standard elements of RSS and Atom feeds.

You can then execute SQL queries against either the RSSFeed or AtomFeed tables. The adapter will retrieve the data from the URL specified in the URL connection string property.

Connect to Feeds with Custom Extensions

To connect to feeds that have custom extensions, you can use the Tables property. In this case, the schema (the columns available for the table) must be dynamically determined from the URL. Therefore, the RSSFeed and AtomFeed tables are not useful. The adapter supports custom feeds by allowing you to create your own tables. You will find a tutorial to accessing nonstandard feeds in Accessing Custom Feeds.

Accessing Custom Feeds

Accessing Nonstandard or Custom Feeds

For most feeds, the RSSFeed and AtomFeed tables outlined in Retrieving Data from Feeds will be sufficient. However, some feeds contain elements that are not standard to the RSS or Atom specifications; for example, the NPR podcast below:

http://www.npr.org/rss/podcast.php?id=510298

Nonstandard elements in this feed like 'itunes:duration' will not appear on the RSSFeed and AtomFeed tables. To access them, you will need to create a new table or modify an existing table to add columns that correspond to the feed elements you want to query.

Create a New Table Schema, Patterned from RSSFeed

To map feed elements to columns, the adapter relies on schema files that define the table. The table schemas are defined in the *.rsd files located in the db subfolder of the installation directory. You can template the definitions for new tables from these default tables.

To create a new table, copy the schema for the RSSFeed table. The name of the file determines the table name. For simplicity, the name of the schema file in this example is PodcastExample.rsd.

Add Custom Columns

Table columns are defined in the *rsb:info* block, which defines the column names, data types, and descriptions. To add a new column to a table, you can simply append an attribute to rsb:info. You will need to set the following values:

name: the desired column name. The column names do not need to match the element names from the feed; you will define this mapping in the next step.

xs:type: an appropriate data type. The adapter accepts the following data types: int, double, datetime, time, string, long, boolean, and decimal.

key: whether this column is a primary key (a unique identifier that can be used to select this record and this record only).

desc: a description for users.

For example, to add the *itunes:duration* field from the example podcast to a column with the name "Duration", add the line below:

```
<rsb:info>
```

```
      ...
     <attr name="Duration"         xs:type="string"    key="false"
desc="The duration of a podcast." />
    </rsb:info>
```

Map the Custom Element to the Column Name

Next, map the actual name found in the feed to the column name. Use the *rsb:set* command in the operation block to define the attribute you are setting, the actual name, and a default value to use if the field is empty. A line can simply be added below the existing sets:

```
    <rsb:script method="GET">
      <rsb:call op="[_connection.url]" ignoreprefix='rss'>
        ...
        <rsb:set attr="Duration"  value="[itunes:duration |
def('')]"/>
        <rsb:push/>
      </rsb:call>
    </rsb:script>
```

Advanced: Map Any Element

Note that the element defined is assumed to be under the repeating <item> element in the feed. In order to map elements located higher up in the feed, the _meta prefix is needed. In this tutorial, the FeedTitle and FeedImage columns use the _meta prefix to access the <title> element from the feed level. This causes the element to be displayed in every entry.

Supported SQL

The RSS Adapter supports several basic operations on data, including querying, deleting, modifying, and inserting. These operations are performed using SQL that is close to standard SQL, but may have minor differences described here.

The adapter does not support batching of SQL statements. To execute multiple commands, create multiple instances and execute each separately.

SELECT Statements

SELECT statements generally follow the standard SQL syntax. For more information and specific examples, see SELECT Statements.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings should be quoted using single quotes (e.g., 'John Doe').

Transactions

Transactions are not currently supported.

SELECT Statements

Features

The following features are supported for the SELECT statement:

Using * to select all columns is supported.

```
SELECT * FROM AtomFeed
```

Column aliases are supported.

```
SELECT Author AS MY_Author FROM AtomFeed
```

Quoted column names are supported.

```
SELECT "Author" FROM AtomFeed
```

The following operators are supported in the WHERE clause: .

```
SELECT * FROM AtomFeed WHERE Category = 'US';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Author INTO "csv://AtomFeed.txt" FROM "AtomFeed" WHERE
Category = US
```

You can specify other formats in the file URI. The following example exports tab-separated values:

```
SELECT * INTO "AtomFeed" IN
"csv://filename=c:/AtomFeed.csv;delimiter=tab" FROM "AtomFeed"
LIMIT 101
```

Data Model

Tables

The RSS Adapter models RSS entities in relational Views, or read-only tables. The RSS Adapter comes with two default Views: RSSFeed and AtomFeed. These views model generic feeds and will return data elements typically found in RSS and Atom feeds. You can access custom data elements by defining your own tables using the Tables connection string property. See Retrieving Data from Feeds and Accessing Custom Feeds for more information.

Views

Views are similar to tables in the way that data is represented; however, views do not support updates. Data sources that are represented as views are typically read-only data sources. Often, a stored procedure is available to update the data if such functionality is applicable to the data source.

Queries can be executed against a view as if it were a normal table, and the data that comes back is similar in that regard. To find out more about tables and stored procedures, please navigate to their corresponding entries in this help document.

Views are composed of columns and pseudo columns.

Data Model

A full list is available upon customer request.

TDV Salesforce with SSO Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Authenticating to Salesforce APIs

There are several authentication methods available for connecting to Salesforce with SSO: Login credentials, SSO, and OAuth.

Authenticating with a Login and Token

Set the User and Password to your login credentials. Additionally, set the SecurityToken. By default, the SecurityToken is required, but you can make it optional by allowing a range of trusted IP addresses:

To disable the security token, log into Salesforce and in the setup section enter "Network Access" in the Quick Find box. Add your IP address to the list of trusted IP addresses.

To obtain the security token, follow the steps below:

Open the personal information page on Salesforce.com.

Click the link to reset your security token. The token will be emailed to you.

Specify the security token in the

SecurityToken connection property or append it to the Password.

Authenticating with SSO

The adapter supports SSO authentication through the OneLogin and PingFederate identity providers. Both identity providers require the following common connection properties. You can find the values for the following properties in the Salesforce account settings by navigating to Administration Setup -> Security Controls -> SAML Single Sign-On Settings and then choosing the desired organization.

SSOLoginUrl: Set this to your SSO endpoint.

Below is an example for OneLogin:
`https://app.onelogin.com/saml/metadata/123455`

Below is an example for PingFederate:
`https://myssoendpoint/idp/sts.wst`

SSOTokenUrl: Set this to your Salesforce with SSO OAuth 2.0 token endpoint for SSO. For example:
`https://mysite.salesforce.com/services/oauth2/token?so=1234567`

You also need to specify other properties specific to the identity provider in SSOProperties. Specify these properties as name-value pairs separated by semicolons. Additionally, set the IdPName property to "OneLogin" or "PingFederate" in SSOProperties.

Authenticating with OAuth

If you do not have access to the username and password or do not wish to require them, you can use the OAuth user consent flow. See See Using OAuth for a guide.

Connecting to Salesforce with SSO APIs

By default, the adapter connects to production environments. Set UseSandbox to true to use a Salesforce with SSO sandbox account. Ensure that you specify a sandbox username in User.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

All Or None	A boolean indicating if you would like all inserts, updates, or deletes to fail in a request if even a single record fails.
API Version	The version of the Salesforce API used.
Bulk Polling Interval	The time interval in milliseconds between requests that check the availability of the bulk query response. The default value is 500 ms.

Bulk Query Timeout	The timeout in minutes for which the provider will wait for a bulk query response. The default value is 25 minutes.
Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Login URL	URL to the Salesforce server used for logging in.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Refresh Token	The OAuth refresh token for the corresponding OAuth access token.
OAuth Server Url	The server URL to use if authenticating with OAuth.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.

Password	The password of the Salesforce account used to authenticate to the Salesforce server.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Salesforce from the provider.
Security Token	The security token used to authenticate access to the Salesforce account.
Session Timeout	The time in minutes for which a Salesforce login session is reused.
SSL Client Cert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSL Client Cert Password	The password for the TLS/SSL client certificate.
SSL Client Cert Subject	The subject of the TLS/SSL client certificate.
SSL Client Cert Type	The type of key store containing the TLS/SSL client certificate.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
SSO Login Url	The identity provider's login URL.
SSO Properties	Additional properties required to connect to the identity provider in a semicolon-separated list.

SSO Token Url	The Salesforce OAuth 2.0 token endpoint for the identity provider.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use Bulk API	Whether to use the synchronous SOAP API or the asynchronous Bulk API.
User	The username of the Salesforce account used to authenticate to the server.
Use Sandbox	A boolean determining if the connection should be made to a Salesforce sandbox account.

All Or None

A boolean indicating if you would like all inserts, updates, or deletes to fail in a request if even a single record fails.

Data Type

bool

Default Value

false

Remarks

The AllOrNone property is not available when using Bulk API requests.

API Version

The version of the Salesforce API used.

Data Type

string

Default Value

"40.0"

Remarks

The Salesforce with SSO API version used by default is 40.0.

Bulk Polling Interval

The time interval in milliseconds between requests that check the availability of the bulk query response. The default value is 500 ms.

Data Type

string

Default Value

"500"

Remarks

The time interval between requests that check the availability of the bulk query response. When UseBulkAPI is set, the adapter starts an asycnchronous job in Salesforce when running a SELECT query. It then waits for the response to be ready by periodically polling the server to check status. This property controls the frequency of polling.

Bulk Query Timeout

The timeout in minutes for which the provider will wait for a bulk query response. The default value is 25 minutes.

Data Type

string

Default Value

"25"

Remarks

The timeout in minutes for which the adapter will wait for a bulk query response. The default value is 25 minutes. When UseBulkAPI is set, the adapter starts an asycnchronous job in Salesforce when running a SELECT query. It then waits for the response to be ready by periodically polling the server to check status. This property controls the total time the adapter will wait for a response.

Note that this property is very different from Timeout. The Timeout is an inactivity timeout that controls the time to wait for any response. This property controls the total length of time to wait for a bulk query to execute. ;

Callback URL

The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.

Data Type

string

Default Value

"http://localhost:33333"

Remarks

During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Salesforce with SSO and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Login URL

URL to the Salesforce server used for logging in.

Data Type

string

Default Value

""

Remarks

URL to the Salesforce with SSO server used for logging in. Defaults to https://login.salesforce.com/services/Soap/c/40.0. This should only be changed if your organization uses a custom login endpoint.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Refresh Token

The OAuth refresh token for the corresponding OAuth access token.

Data Type

string

Default Value

""

Remarks

The OAuthRefreshToken property is used to refresh the OAuthAccessToken when using OAuth authentication.

OAuth Server Url

The server URL to use if authenticating with OAuth.

Data Type

string

Default Value

""

Remarks

If authenticating with OAuth, do not enter the User, Password, and SecurityToken.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\Salesforce Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password of the Salesforce account used to authenticate to the Salesforce server.

Data Type

string

Default Value

""

Remarks

Together with User and SecurityToken, this field is used to authenticate against the Salesforce with SSO server.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Salesforce from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Security Token

The security token used to authenticate access to the Salesforce account.

Data Type

string

Default Value

""

Remarks

Together with User and Password, this field can be used to authenticate against the Salesforce with SSO server. This is only required if your organization is set up to require it. A security token can be obtained by going to your profile information and resetting your security token. If your password is reset, you will also need to reset the security token.

Session Timeout

The time in minutes for which a Salesforce login session is reused.

Data Type

string

Default Value

"10"

Remarks

The adapter creates a login session with Salesforce based on the authentication credentials provided. This session is reused for subsequent queries until it times out. The SessionTimeout property allows you to control how long the login session is kept by the adapter. You can set the SessionTimeout to 0 to disable login sessions; this will force the adapter to authenticate each request.

SSL Client Cert

The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

Data Type

string

Default Value

""

Remarks

The name of the certificate store for the client certificate.

The SSLClientCertType field specifies the type of the certificate store specified by SSLClientCert. If the store is password protected, specify the password in SSLClientCertPassword.

SSLClientCert is used in conjunction with the SSLClientCertSubject field in order to specify client certificates. If SSLClientCert has a value, and SSLClientCertSubject is set, a search for a certificate is initiated. Please refer to the SSLClientCertSubject field for details.

Designations of certificate stores are platform-dependent.

The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys.

CA	Certifying authority certificates.
ROOT	Root certificates.
SPC	Software publisher certificates.

In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

SSL Client Cert Password

The password for the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

SSL Client Cert Subject

The subject of the TLS/SSL client certificate.

Data Type

string

Default Value

"*"

Remarks

When loading a certificate the subject is used to locate the certificate in the store.

If an exact match is not found, the store is searched for subjects containing the value of the property.

If a match is still not found, the property is set to an empty string, and no certificate is selected.

The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance "CN=www.server.com, OU=test, C=US, E=support@cdata.com". Common fields and their meanings are displayed below.

Field	Meaning
CN	Common Name. This is commonly a host name like www.server.com.
O	Organization
OU	Organizational Unit
L	Locality
S	State
C	Country
E	Email Address

If a field value contains a comma it must be quoted.

SSL Client Cert Type

The type of key store containing the TLS/SSL client certificate.

Data Type

string

Default Value

""

Remarks

This property can take one of the following values:

USER - default	For Windows, this specifies that the certificate store is a certificate store owned by the current user. *Note*: This store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine store. *Note*: this store type is not available in Java.
PFXFILE	The certificate store is the name of a PFX (PKCS12) file containing certificates.
PFXBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in PFX (PKCS12) format.
JKSFILE	The certificate store is the name of a Java key store (JKS) file containing certificates. *Note*: this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in Java key store (JKS) format. *Note*: this store type is only available in Java.
PEMKEY_FILE	The certificate store is the name of a PEM-encoded file that contains a private key and an optional certificate.
PEMKEY_BLOB	The certificate store is a string (base64-encoded) that contains a private key and an optional certificate.
PUBLIC_KEY_FILE	The certificate store is the name of a file that contains a PEM- or DER-encoded public key certificate.
PUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains a PEM- or DER-encoded public key certificate.
SSHPUBLIC_KEY_FILE	The certificate store is the name of a file that contains an SSH-style public key.

SSHPUBLIC_KEY_BLOB	The certificate store is a string (base-64-encoded) that contains an SSH-style public key.
P7BFILE	The certificate store is the name of a PKCS7 file containing certificates.
PPKFILE	The certificate store is the name of a file that contains a PPK (PuTTY Private Key).
XMLFILE	The certificate store is the name of a file that contains a certificate in XML format.
XMLBLOB	The certificate store is a string that contains a certificate in XML format.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----

A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

SSO Login Url

The identity provider's login URL.

Data Type

string

Default Value

""

Remarks

The identity provider's login URL. Can be found in the Salesforce account settings by opening Administration Setup -> Security Controls -> Single Sign-On Settings and then choosing the desired organization.

SSO Properties

Additional properties required to connect to the identity provider in a semicolon-separated list.

Data Type

string

Default Value

""

Remarks

Additional properties required to connect to the identity provider in a semicolon-separated list. SSOProperties is used in conjunction with the SSOLoginUrl and SSOTokenUrl. The following sections provide examples using the OneLogin and PingFederated identity providers.

OneLogin

The following SSOProperties are needed to authenticate to OneLogin:

IdPName: Set this to OneLogin.

OneLogin API Key: Set this to the API Key, which can be obtained by clicking Settings -> API -> View Legacy API Key.

The following connection string uses an API key to connect to OneLogin:

```
SSOLoginURL=https://app.onelogin.com/saml/metadata/123455;SSO
Token
Url=https://mysite.salesforce.com/services/oauth2/token?so=1234567
;SSOProperties='IdPName=OneLogin;APIKey=MyAPIKey';
```

PingFederate

The following SSOProperties are needed to authenticate to PingFederate.

IdPName: Set this to PingFederate.

IdPSystemScheme: The authorization scheme to be used for the IdP endpoint. The allowed values for this IdP are None or Basic.

IdPSystemUserName: Set this to an IdP user if authentication is to be used for the IdP endpoint.

IdPSystemPassword: Set this to the password of the IdP if authentication is to be used for the IdP endpoint.

Additionally, you can use the following properties to configure mutual SSL authentication for the SSOLoginUrl, the WS-Trust STS endpoint:

IdPSystemSSLClientCert: Behaves just like SSLClientCert

IdPSystemSSLClientCertType: Behaves just like

SSLClientCertType

IdPSystemSSLClientCertSubject: Behaves just like

SSLClientCertSubject

IdPSystemSSLClientCertPassword: Behaves just like

SSLClientCertPassword

Below is an example connection string:

```
SSO Login Url=https://myssoendpoint/idp/sts.wst;SSO Token
Url=https://mysite.salesforce.com/services/oauth2/token?so=1234567
;SSO Properties ='IdPName=pingfederate;IdPSystemScheme=basic;
IdPSystemUsername=Administrator; IdPSystemPassword=xA123456;'"
```

SSO Token Url

The Salesforce OAuth 2.0 token endpoint for the identity provider.

Data Type

string

Default Value

""

Remarks

The Salesforce OAuth 2.0 token endpoint for the identity provider. This can be found in the Salesforce account settings by navigating to Administration Setup -> Security Controls -> SAML Single Sign-On Settings and then choosing the desired organization.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use Bulk API

Whether to use the synchronous SOAP API or the asynchronous Bulk API.

Data Type

bool

Default Value

false

Remarks

Whether or not the asynchronous Bulk API is used for reads and writes to Salesforce with SSO. For reads, the driver will automatically create bulk query jobs and start returning results as they are available. Note that queries using a JOIN or aggregation are not supported in the Bulk API and therefore the driver will fall back to using the SOAP API for these queries. Jobs are closed automatically once they are complete or if there is a failure.

For writes back to Salesforce with SSO, up to 10000 records can be sent per batch. These requests will be processed asynchronously meaning the driver will not wait for Salesforce to fully process the results. You can query the following table to get information about the jobs and batches that were created:

```
SELECT * FROM "Info#TEMP"
```

The id values returned from this can be used with GetJob, GetBatch, and GetBatchResults to check the status of the job and batches.

Note that when this property is set to false, the default, you can still modify multiple records simultaneously: The adapter will perform batch processing through the Web Services API, which will synchronously return the status of your operations. When you invoke the JDBC Batch API, the adapter will batch multiple requests together in a single request to Salesforce with SSO.

User

The username of the Salesforce account used to authenticate to the server.

Data Type

string

Default Value

""

Remarks

Together with Password and SecurityToken, this field is used to authenticate to the Salesforce server.

Use Sandbox

A boolean determining if the connection should be made to a Salesforce sandbox account.

Data Type

bool

Default Value

false

Remarks

In order to connect to a Salesforce sandbox account, set the UseSandbox to TRUE and append the sandbox name to the end of the username. For instance, if your username is "user", and sandbox name in Salesforce is "sandbox", the specified User should appear as "user.sandbox".

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.
`Verbosity=4;`

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Salesforce with SSO Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Using OAuth

OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Configuring the Adapter

The adapter can be configured for OAuth in three ways using the InitiateOAuth property.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and Salesforce with SSO.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

CallbackURL

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from Salesforce with SSO and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If Salesforce with SSO issues a long-lived access token, use the Salesforce with SSO developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

Advanced Settings

Processing Bulk Jobs Asynchronously

By default the adapter uses the Salesforce with SSO Web Services APIs, which have synchronous batch processing functionality; that is, you can modify multiple records simultaneously and synchronously.

The asynchronous Bulk API makes large requests easier because execution does not stop while the adapter waits to be given back control when the call returns.

See the UseBulkAPI property for more information.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the connection properties needed to authenticate and to connect. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Specifying an API Version

You can set APIVersion forwards or backwards to use different versions of the Web Services API.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

SQL Compliance

The Salesforce with SSO Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Salesforce with SSO API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

UPSERT Statements

An UPSERT updates a record if it exists and inserts the record if it does not. See UPSERT Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

CREATE TABLE Statements

See CREATE TABLE Statements for a syntax reference and examples.

DROP TABLE Statements

See DROP TABLE Statements for a syntax reference and examples.

ALTER TABLE Statements

See ALTER TABLE Statements for a syntax reference and examples.

GETDELETED Statements

GETDELETED statements return the Ids of deleted records. See GETDELETED Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Salesforce with SSO adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  } [ , ... ]
  [ [
      INNER | { { LEFT | RIGHT | FULL  } [ OUTER ] }
    ] JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [ HAVING <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | NOT LIKE
| IS NULL | IS NOT NULL | IN | NOT IN | AND | OR } [ <expression> ]
```

```
} [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Account
```

Rename a column:
```
SELECT "Name" AS MY_Name FROM Account
```

Search data:
```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

The Salesforce with SSO APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, NOT LIKE, IS NULL, IS NOT NULL, IN, NOT IN, AND, OR.
```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

Return the number of items matching the query criteria:
```
SELECT COUNT(*) AS MyCount FROM Account
```

Return the number of unique items matching the query criteria:
```
SELECT COUNT(DISTINCT Name) FROM Account
```

Summarize data:
```
SELECT Name, MAX(AnnualRevenue) FROM Account  GROUP BY Name
```

See Aggregate Functions for details.

Retrieve data from multiple tables.
```
SELECT Account.Id, Account.Name, Account.Fax,
Opportunity.AccountId, Opportunity.CloseDate FROM Account INNER
JOIN Opportunity ON Account.Id = Opportunity.AccountId
```

See JOIN Queries for details.

Sort a result set in ascending order:
```
SELECT BillingState, Name FROM Account  ORDER BY Name ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM Account WHERE Industry = 'Floppy Disks'
```

COUNT_DISTINCT

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT_DISTINCT(BillingState) AS DistinctValues FROM Account
WHERE Industry = 'Floppy Disks'
```

AVG

Returns the average of the column values.

```
SELECT Name, AVG(AnnualRevenue) FROM Account WHERE Industry =
'Floppy Disks' GROUP BY Name
```

MIN

Returns the minimum column value.

```
SELECT MIN(AnnualRevenue), Name FROM Account WHERE Industry =
'Floppy Disks' GROUP BY Name
```

MAX

Returns the maximum column value.

```
SELECT Name, MAX(AnnualRevenue) FROM Account WHERE Industry =
'Floppy Disks' GROUP BY Name
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(AnnualRevenue) FROM Account WHERE Industry = 'Floppy
Disks'
```

JOIN Queries

This section provides information about the features and restrictions that are specific to how Salesforce supports joins. The Salesforce with SSO Adapter supports join queries based on SOQL (Salesforce Object Query Language). The adapter supports standard SQL syntax instead of proprietary SOQL to allow easy integration with a wide variety of SQL tools. Join queries in Salesforce are based on the relationships among Salesforce objects.

Relationship Queries

Salesforce objects can be linked using relationships. The standard Salesforce objects have predefined relationships. You can define relationships for your custom objects.

Parent to Child Relationships

Salesforce relationships are directional and are of the following types: one-to-many (parent to child) or many-to-one (child to parent). Since the relationships are directional the order in which the tables are included in the query determines the path of relationship traversal.

The following query shows a simple parent-to-child join query. This query returns all Accounts and the first and last name of each Contact associated with that Account.

```
SELECT Contact.FirstName, Account.Name
FROM Account, Contact
```

Polymorphic Relationships

Salesforce relationships can be polymorphic. That is, a given relationship on a field can refer to more than one type of entity. For example, the Task entity contains a Who relationship, which by default may refer to a Contact or Lead.

The following query shows a join based on a polymorphic relationship. To return only Tasks referring to a Contact on the Who relationship, specify the type of entity in the relationship:

```
SELECT Task.Subject, Contact.Name
FROM Task, Contact
WHERE Contact.Type='Contact'
```

Custom Relationships

You can specify a join condition that is a custom relationship. The following query retrieves the names of all Account records and the first names of all Contacts that match the specified join condition:

```
Select Contact.Firstname, Account.Name
FROM Account
JOIN Contact
ON Account.MyCustomColumn__c = Contact.Id
```

Supported Join Syntax

You can use the syntax detailed below to execute joins on Salesforce objects. You can only join related objects.

A benefit of predefined relationships is that you do not need to specify the join conditions to execute a join with Salesforce data; the conditions are already accounted for based on the relationship. The following query returns the first names of all the Contacts in the organization and for each Contact the name of the parent Account associated with that Contact.

```
Select Contact.Firstname, Account.Name
FROM Contact, Account
```

If there are multiple relationships between the tables, you can explicitly set the join criteria. The following query matches on a custom relationship determined by the columns in the join criteria (Account.MyCustomColumn__c and Contact.Id), instead of the default parent-to-child relationship between Accounts and Contacts:

```
Select Contact.Firstname, Account.Name
FROM Account
JOIN Contact
ON Account.MyCustomColumn__c = Contact.Id
```

Inner joins are supported. The following query retrieves all Account records that are associated with an Opportunity:

```
SELECT Account.Id, Account.Name, Account.Fax,
Opportunity.AccountId, Opportunity.CloseDate
FROM Account
INNER JOIN Opportunity
ON Account.Id = Opportunity.AccountId
```

The following query shows a join between three custom objects. This query joins the custom tables NW_Product__c, NW_Category__c, and NW_Suppliers__c and returns the Name field from each of them. Additionally, the results from the NW_Category__c table are filtered for names that start with "Dairy":

```
SELECT A.Name, B.Name, C.Name
FROM  NW_Product__c as A, NW_Category__c as B, NW_Suppliers__c as C

WHERE B.Name LIKE 'Dairy%'
```

Projection Functions

CONVERTCURRENCY(column)

Returns the currency field converted to the user's currency

column: Any column expression.

CALENDAR_MONTH(column)

Returns a number representing the calendar month of a date field (1 for January, 12 for December).

column: Any column expression.

CALENDAR_QUARTER(column)

Returns a number representing the calendar quarter of a date field (1 for January 1 through March 31, 2 for April 1 through June 30, 3 for July 1 through September 30, 4 for October 1 through December 31).

column: Any column expression.

CALENDAR_YEAR(column)

Returns a number representing the calendar year of a date field (2009).

column: Any column expression.

DAY_IN_MONTH(column)

Returns a number representing the day in the month of a date field (20 for February 20).

column: Any column expression.

DAY_IN_WEEK(column)

Returns a number representing the day of the week for a date field (1 for Sunday, 7 for Saturday).

column: Any column expression.

DAY_IN_YEAR(column)

Returns a date representing the day portion of a dateTime field (32 for February 1).

column: Any column expression.

DAY_ONLY(column)

Returns a date representing the day portion of a dateTime field (2009-09-22 for September 22, 2009).

column: Any column expression.

FISCAL_MONTH(column)

Returns a number representing the fiscal month of a date field. This differs from CALENDAR_MONTH() if your organization uses a fiscal year that does not match the Gregorian calendar. If your fiscal year starts in March: 1 for March, 12 for February.

column: Any column expression.

FISCAL_QUARTER(column)

Returns a number representing the fiscal quarter of a date field. This differs from CALENDAR_QUARTER() if your organization uses a fiscal year that does not match the Gregorian calendar. If your fiscal year starts in July: 1 for July 15, 4 for June 6.

column: Any column expression.

FISCAL_YEAR(column)

Returns a number representing the fiscal year of a date field. This differs from CALENDAR_YEAR() if your organization uses a fiscal year that does not match the Gregorian calendar (2009).

column: Any column expression.

HOUR_IN_DAY(column)

Returns a number representing the hour in the day for a dateTime field (18 for a time of 18:23:10).

column: Any column expression.

WEEK_IN_MONTH(column)

Returns a number representing the week in the month for a date field (2 for April 10). The first week is from the first through the seventh day of the month.

column: Any column expression.

WEEK_IN_YEAR(column)

Returns a number representing the week in the year for a date field (1 for January 3). The first week is from January 1 through January 7.

column: Any column expression.

Predicate Functions

CONVERTCURRENCY(column)

Returns the currency field converted to the user's currency

column: Any column expression.

CALENDAR_MONTH(column)

Returns a number representing the calendar month of a date field (1 for January, 12 for December).

column: Any column expression.

CALENDAR_QUARTER(column)

Returns a number representing the calendar quarter of a date field (1 for January 1 through March 31, 2 for April 1 through June 30, 3 for July 1 through September 30, 4 for October 1 through December 31).

column: Any column expression.

CALENDAR_YEAR(column)

Returns a number representing the calendar year of a date field (2009).

column: Any column expression.

DAY_IN_MONTH(column)

Returns a number representing the day in the month of a date field (20 for February 20).

column: Any column expression.

DAY_IN_WEEK(column)

Returns a number representing the day of the week for a date field (1 for Sunday, 7 for Saturday).

column: Any column expression.

DAY_IN_YEAR(column)

Returns a date representing the day portion of a dateTime field (32 for February 1).

column: Any column expression.

DAY_ONLY(column)

Returns a date representing the day portion of a dateTime field (2009-09-22 for September 22, 2009).

column: Any column expression.

FISCAL_MONTH(column)

Returns a number representing the fiscal month of a date field. This differs from CALENDAR_MONTH() if your organization uses a fiscal year that does not match the Gregorian calendar. If your fiscal year starts in March: 1 for March, 12 for February.

column: Any column expression.

FISCAL_QUARTER(column)

Returns a number representing the fiscal quarter of a date field. This differs from CALENDAR_QUARTER() if your organization uses a fiscal year that does not match the Gregorian calendar. If your fiscal year starts in July: 1 for July 15, 4 for June 6.

column: Any column expression.

FISCAL_YEAR(column)

Returns a number representing the fiscal year of a date field. This differs from CALENDAR_YEAR() if your organization uses a fiscal year that does not match the Gregorian calendar (2009).

column: Any column expression.

HOUR_IN_DAY(column)

Returns a number representing the hour in the day for a dateTime field (18 for a time of 18:23:10).

column: Any column expression.

WEEK_IN_MONTH(column)

Returns a number representing the week in the month for a date field (2 for April 10). The first week is from the first through the seventh day of the month.

column: Any column expression.

WEEK_IN_YEAR(column)

Returns a number representing the week in the year for a date field (1 for January 3). The first week is from January 1 through January 7.

column: Any column expression.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT BillingState, Name INTO "csv://Account.txt" FROM "Account"
WHERE Industry = 'Floppy Disks'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT BillingState, Name INTO "csv://Account.txt;delimiter=tab"
FROM "Account" WHERE Industry = 'Floppy Disks'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Account SET Name='John' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1045625d-99ee-e011-a272-00155d01ad6b");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

UPSERT Statements

An UPSERT statement will update an existing record or create a new record if an existing record is not identified.

Configuring Upserts

Upserts can only be performed on a field explicitly defined in Salesforce to be an external key. You will need to provide the name of this field in a column called ExternalIdColumn, as in the query below:

```
  UPSERT INTO Lead (FirstName, LastName, Company,
External_Id_Column__c, ExternalIdColumn)
  VALUES ('Bob', 'Thorton', 'Universal Pictures', 12345,
'External_Id_Column__c')
```

In order for the ExternalIdColumn to show up, modify the Salesforce connection properties to set the PseudoColumns field to the value '*=*' without the quotes. An UPSERT statement will update an existing record or create a new record if an existing record is not identified.

UPSERT Syntax

The UPSERT syntax is the same as for insert. Salesforce with SSO uses the input provided in the VALUES clause to determine whether the record already exists. If the record does not exist, all columns required to insert the record must be specified. See Data Model for any table-specific information.

```
UPSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "UPSERT INTO Account (Name) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "John");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:salesforce:User=myUser;Password=
myPassword;Security Token=myToken;",);
String cmd = "DELETE FROM Account WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1045625d-99ee-e011-a272-00155d01ad6b");
int count=pstmt.executeUpdate();
connection.close();
```

GETDELETED Statements

You can issue the GETDELETED query to retrieve all records deleted from the live data for the time range specified. This query accepts a datetime value as a filter.

```
GETDELETED FROM <table_name> WHERE <search_condition>

<search_condition> ::=
  {
    <expression> { = | < | <= | > | >= } [ <expression> ]
  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

```
String cmd = "GETDELETED FROM Account WHERE TimeModified
>='2013-01-01' AND TimeModified <='2013-02-01'";
PreparedStatement pstmt = connection.prepareStatement(cmd);
ResultSet rs = pstmt.getResultSet();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

CREATE TABLE Statements

To create new Salesforce with SSO entities, use CREATE TABLE statements:

CREATE TABLE Syntax

The CREATE TABLE statement specifies the table name and a comma-separated list of column names and the primary keys of the table.

```
CREATE TABLE <table_name> IF [ NOT EXISTS ]
(
  {
     <column_name> <data_type>
     [ NOT NULL ]
     [ DEFAULT <literal> ]
     [ PRIMARY KEY ]
     [ UNIQUE ]
  } |  PRIMARY KEY ( <column_name> [ , ... ] )
  [ , ... ]
)
```

Example Query:

The following statement creates a MyCustomers table on the Salesforce with SSO server with name, age, and address columns.

```
CREATE TABLE IF NOT EXISTS "MyCustomers" (name VARCHAR(20), age
INT, address VARCHAR(20))
```

DROP TABLE Statements

Use DROP TABLE statements to delete a table and all the data it contains from Salesforce with SSO.

DROP TABLE Syntax

The DROP TABLE statement accepts the name of the table to delete.

```
DROP TABLE [ IF EXISTS ] <table_name>
```

Example Query:

The following query deletes all MyCustomers data from the server.

```
DROP TABLE IF EXISTS MyCustomers
```

ALTER TABLE Statements

Use the ALTER TABLE statement to add, delete, or modify the columns of a table.

ALTER TABLE Syntax

To add, delete, or modify columns, use the ADD, ALTER, or DROP keywords of the ALTER TABLE statement. The ADD keyword accepts a column definition or a comma-separated list of column definitions. The ALTER keyword accepts a column definition. The DROP keyword accepts a column name.

```
ALTER TABLE <table_name>
  ADD [ COLUMN ] [ IF NOT EXISTS ]
    <column_definition> | ( <column_definition> [ , ... ] )
|ALTER COLUMN <column_definition>
|DROP COLUMN [ IF EXISTS ] <column_name>

<column_definition> ::=
  <column_name>
  <data_type>
  [ NOT NULL ]
  [ DEFAULT <literal> ]
  [ PRIMARY KEY ]
  [ UNIQUE ]
```

Example Queries:

The following query adds a new ExternalCustomerId column on the server:

```
ALTER TABLE MyCustomers ADD (ExternalCustomerId int)
```

The following query changes the data type of the ExternalCustomerId column:

```
ALTER TABLE MyCustomers ALTER COLUMN ExternalCustomerId  string
```

The following query removes the ExternalCustomerId column:

```
ALTER TABLE MyCustomers DROP COLUMN  ExternalCustomerId
```

Data Model

A full list is available upon customer request.

TDV SharePoint Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Connecting to SharePoint

Set the URL to a Site Collection to work with all Lists and Documents in all nested Subsites. Set the URL to a specific Site to work with Lists and Documents in that Site only.

URL	Example URL
Site	https://teams.contoso.com/teamA or https://teamA.contoso.com
Site Collection	https://teams.contoso.com

In addition to providing the URL, use one of the following sets of connection properties to authenticate to SharePoint. The default values make it easy to connect in most environments, as shown below.

Authenticating to SharePoint Online

Set SharePointEdition to "SharePoint Online" and set the User and Password to the credentials you use to log onto SharePoint; for example, the credentials to your Microsoft Online Services account.

The following SSO (single sign-on) identity providers are also supported: Azure Active Directory, OneLogin, and OKTA.

SSO

If you have enabled SSO in SharePoint Online, set UseSSO to true in addition to your User and Password.

If the user account domain is different from the domain configured with the identity provider, set SSODomain to the domain configured with the identity provider. This property may be required for AD FS, OneLogin, and OKTA.

Authenticating to SharePoint On Premises

Set SharePointEdition to "SharePoint OnPremise" to use the following authentication types.

Windows (NTLM)

This is the most common authentication type. As such, the adapter is preconfigured to use NTLM as the default; simply set the Windows User and Password to connect.

Kerberos and Kerberos Delegation

To authenticate with Kerberos, set AuthScheme to NEGOTIATE. If needed, provide the User and Password. To use Kerberos Delegation, set AuthScheme to KERBEROSDELEGATION.

KerberosKDC, KerberosSPN, and KerberosRealm enable control over the components of Kerberos authentication.

Forms

This allows authentication through a custom authentication method, instead of Active Directory. To use this authentication type, set AuthScheme to FORMS and set the User and Password.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Cookie	The cookie to use for authenticating to the online SharePoint server.

Auth Scheme	The scheme used for authenticating to SharePoint On-Premise instances. Accepted entries are NTLM, BASIC, DIGEST, FORMS, NONE, NEGOTIATE, and KERBEROSDELEGATION. NTLM is the default.
Calculated Data Type	The data type to be used for calculated fields.
Continue On Error	Indicates whether or not to continue updating items in a batch after an error.
Create ID Columns	Indicates whether or not to create supplemental ID columns for SharePoint columns that use values from information stored in other Lists.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Folder Option	An option to determine how to display folders in results. Enter either FilesOnly, FilesAndFolders, Recursive, or RecursiveAll.
Kerberos KDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
Kerberos Realm	The Kerberos Realm used to authenticate the user with.
Kerberos SPN	The Service Principal Name for the Kerberos Domain Controller.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Page Size	The number of results to return per page of data retrieved from SharePoint.
Password	The password used to authenticate the user.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.

Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to SharePoint from the provider.
Share Point Edition	The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.
Show Hidden Columns	Boolean determining if hidden columns should be shown or not. If false, all hidden columns will be removed from the column listing.
Show Predefined Columns	Boolean determining if predefined columns should be shown or not. If false, all columns derived from a base type will be removed from the column listing.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
SSO Domain	The domain of the user when using single sign-on (SSO).
STSURL	The URL of the security token service (STS) when using single sign-on (SSO).
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
URL	The base URL for the site.
Use Display Names	Boolean determining if the display names for the columns should be used instead of the API names.
User	The SharePoint user account used to authenticate.

Use Simple Names	Boolean determining if simple names should be used for tables and columns.
Use SSO	Whether or not to use single sign-on (SSO) to authenticate to SharePoint Online.

Auth Cookie

The cookie to use for authenticating to the online SharePoint server.

Data Type

string

Default Value

""

Remarks

After logging in to http://*yourdomain*.sharepoint.com/TeamSite, the "keep me signed in" setting must be checked in order for the connection to be established.

Auth Scheme

The scheme used for authenticating to SharePoint On-Premise instances. Accepted entries are NTLM, BASIC, DIGEST, FORMS, NONE, NEGOTIATE, and KERBEROSDELEGATION. NTLM is the default.

Data Type

string

Default Value

"NTLM"

Remarks

Together with Password and User, this field is used to authenticate against the server. NTLM is the default option. Use the following options to select your authentication scheme:

NTLM: Set this to use your Windows credentials for authentication.

NEGOTIATE: If

AuthScheme is set to NEGOTIATE, the adapter will negotiate an authentication mechanism with the server. Set AuthScheme to NEGOTIATE if you want to use Kerberos authentication.

KERBEROSDELEGATION: Set this to use delegation through the Kerberos protocol. Set the

User and Password of the account you want to impersonate.

NONE: Set this to use anonymous authentication; for example, to access a public site.

FORMS: Set this if your SharePoint instance uses a custom authentication method through a Web form.

DIGEST: Set this to use HTTP Digest authentication.

BASIC: Set this to use HTTP Basic authentication.

Calculated Data Type

The data type to be used for calculated fields.

Data Type

string

Default Value

""

Remarks

The data type to be used for calculated fields. By default, the data type is determined by the type of calculated field in SharePoint. However, in some cases these calculated fields may return values that are not appropriate for the specified type. In these instances, you may wish to set the Calculated Data Type to String.

Continue On Error

Indicates whether or not to continue updating items in a batch after an error.

Data Type

bool

Default Value

true

Remarks

If this property is set to True (default), the adapter will continue adding, updating, or deleting items when an error is encountered on one of the items. When set to False, the adapter will stop adding, updating, or deleting items after an error is encountered (entries preceeding the problematic entry will still be added, updated, or deleted).

Create ID Columns

Indicates whether or not to create supplemental ID columns for SharePoint columns that use values from information stored in other Lists.

Data Type

bool

Default Value

true

Remarks

Indicates whether or not to create supplemental ID columns for SharePoint columns that use values from information stored in other Lists (like "Lookup" or "Person or Group" columns). The ID column that is created will contain the related entry's ID (in the context of its original List). If set to false, the ID columns will not be created, the ID will be ignored, and only the value of the referenced column will be returned.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to SharePoint and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.

SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Folder Option

An option to determine how to display folders in results. Enter either FilesOnly, FilesAndFolders, Recursive, or RecursiveAll.

Data Type

string

Default Value

"RecursiveAll"

Remarks

An option to determine how to display folders in results. FilesOnly will display only files in specified lists or libraries. FilesAndFolders will display files and folders in the specified list. RecursiveAll will display files in the specified list and all subfolders.

Kerberos KDC

The Kerberos Key Distribution Center (KDC) service used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos Key Distribution Center (KDC) service. The Kerberos Key Distribution Center (KDC) service is conventionally colocated with the domain controller. If Kerberos KDC is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using the config file krb5.conf, or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the configured domain name and host as a last resort.

Note: Windows authentication is supported in JRE 1.6 and above only.

Kerberos Realm

The Kerberos Realm used to authenticate the user with.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using SPNEGO or Windows Authentication. The Kerberos Realm is used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name. If Kerberos Realm is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using a config file (krb5.conf) or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC

are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the user-configured domain name and host as a last resort. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

Kerberos SPN

The Service Principal Name for the Kerberos Domain Controller.

Data Type

string

Default Value

""

Remarks

If the Service Principal Name on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, set the Service Principal Name here.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Page Size

The number of results to return per page of data retrieved from SharePoint.

Data Type

string

Default Value

"1000"

Remarks

The number of results to return per page of data retrieved from SharePoint. Higher page sizes will result in fewer requests, but timeouts may occur.

Password

The password used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The User, Password, and AuthScheme are together used to authenticate with the server.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to SharePoint from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Share Point Edition

The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.

Data Type

string

Default Value

"SharePoint OnPremise"

Remarks

The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.

Show Hidden Columns

Boolean determining if hidden columns should be shown or not. If false, all hidden columns will be removed from the column listing.

Data Type

bool

Default Value

false

Remarks

Boolean determining if hidden columns should be shown or not. If false, all hidden columns will be removed from the column listing.

Show Predefined Columns

Boolean determining if predefined columns should be shown or not. If false, all columns derived from a base type will be removed from the column listing.

Data Type

bool

Default Value

true

Remarks

Boolean determining if predefined columns should be shown or not. If false, all columns derived from a base type will be removed from the column listing. These columns are normally system columns such as CreatedBy and Author. But, predefined columns may also include common columns such as Title.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801c bb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

SSO Domain

The domain of the user when using single sign-on (SSO).

Data Type

string

Default Value

""

Remarks

This property is only applicable when using single sign-on (UseSSO is set to true) and if the domain of the User (e.g. user@mydomain.com) is different than the domain configured within the SSO service (e.g. user@myssodomain.com).

This property may be required when using the AD FS, OneLogin, or OKTA SSO services.

STSURL

The URL of the security token service (STS) when using single sign-on (SSO).

Data Type

string

Default Value

""

Remarks

The URL of the security token service (STS) when using single sign-on (SSO). This rarely needs to be set explicitly.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

URL

The base URL for the site.

Data Type

string

Default Value

""

Remarks

The following are examples of valid URLs:

http://server/SharePoint/

http://server/Sites/mysite/

http://server:90/

The provider will use URL to derive URLs for other calls to the server.

Use Display Names

Boolean determining if the display names for the columns should be used instead of the API names.

Data Type

bool

Default Value

true

Remarks

Boolean determining if the display names for the columns should be used instead of the API names.

User

The SharePoint user account used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with Password, this field is used to authenticate against the SharePoint server.

For SharePoint On-Premise, User should include the domain and will look similar to the following: DOMAIN\Username.

For SharePoint Online, User will look similar to the following: username@domain.onmicrosoft.com.

Use Simple Names

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

Boolean determining if simple names should be used for tables and columns. SharePoint lists can have special characters in names that are normally not allowed in standard databases. UseSimpleNames makes the adapter easier to use with traditional database tools.

Setting UseSimpleNames to true will simplify the names of tables and columns returned. If set to false, the tables and columns will appear as they do in SharePoint.

Use SSO

Whether or not to use single sign-on (SSO) to authenticate to SharePoint Online.

Data Type

bool

Default Value

false

Remarks

When set to true, single sign-on (SSO) will be used to authenticate to SharePoint Online using the account specified via User and Password. The Active Directory Federation Services (AD FS), OneLogin, and OKTA SSO identity providers are supported.

SSODomain may be required to be set if the domain configured on the SSO domain is different than the domain of the User.

SSO is only applicable when using SharePoint Online. SSO is not supported for On-Premise versions of SharePoint.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the SharePoint Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Fine Tuning the SharePoint Connection

To make it easier to access data in advanced integrations, use the following connection properties to control column name identifiers and other aspects of data access:

UseDisplayNames: Set this to true to return column names that match field names in the underlying API. By default, the adapter uses column names that match the field names defined in SharePoint.

UseSimpleNames: Set this to true to perform substitutions on special characters in column names that SharePoint allows but that many databases typically do not.

ShowPredefinedColumns: Set this to false to exclude fields derived from fields in the list; for example, Author and CreatedAt. This setting excludes the predefined fields from being returned in *SELECT* * statements and schema discovery.

ShowHiddenColumns: When true, columns marked as hidden in SharePoint will be displayed by the adapter.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the SharePoint authentication properties and the URL. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain).

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The SharePoint Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the SharePoint API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the SharePoint adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | != | <> | > | < | >= | <= | BEGINSWITH |
CONTAINS | IN | IS NULL | IS NOT NULL | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Calendar
```

Rename a column:
```
SELECT "Location" AS MY_Location FROM Calendar
```

Search data:
```
SELECT * FROM Calendar WHERE Location <> 'Chapel Hill';
```

The SharePoint APIs support the following operators in the WHERE clause: =, !=, <>, >, <, >=, <=, BEGINSWITH, CONTAINS, IN, IS NULL, IS NOT NULL, AND, OR.
```
SELECT * FROM Calendar WHERE Location <> 'Chapel Hill';
```

Sort a result set in ascending order:
```
SELECT Id, Location FROM Calendar  ORDER BY Location ASC
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):
```
SELECT Id, Location INTO "csv://Calendar.txt" FROM "Calendar" WHERE
Location = 'Chapel Hill'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:
```
SELECT Id, Location INTO "csv://Calendar.txt;delimiter=tab" FROM
"Calendar" WHERE Location = 'Chapel Hill'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Calendar (Location) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "U.S.A.");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause.

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ ,
... ] WHERE { Id = <expression>  } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "UPDATE Calendar SET Location='U.S.A.' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:sharepoint:User=MyUserAccount;Pa
ssword=MyPassword;Auth
Scheme=NTLM;URL=http://sharepointserver/mysite;",);
String cmd = "DELETE FROM Calendar WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

EXECUTE my_proc second = @p1, first = @p2, third = @p3;

Data Model

A full list is available upon customer request.

TDV SharePoint Excel Services Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Details for connecting to SharePoint Online and SharePoint On Premises are below.

Connecting to a Workbook

Regardless of which edition of SharePoint you are using, set File to the Excel workbook. This path is relative to the following properties:

Library: The Shared Documents library is used by default. You can use this property to specify another document library in your organization; for example, if you want to connect to OneDrive for Business, set this property to "Documents".

Folder: You can use this property to specify a path to a subfolder in a library. The path is relative to the library name specified in Library.

Connecting to Spreadsheet Data as Tables

The adapter detects the available tables based on the available objects in the underlying API.

The APIs surface different API objects; select the API based on the organization of your spreadsheets and your SharePoint version:

OData

The OData API allows access to only Excel table objects, which you create by clicking Insert -> Table in Excel. This is the default API.

REST

The REST API enables access to tables defined from Excel table objects, ranges, and spreadsheets. Set UseRESTAPI to true to connect to spreadsheets or ranges as tables; otherwise, the adapter may not return any tables.

By default, the adapter detects column names from the first row; set Header to disable this.

With DefineTables additionally set, you can define tables based on ranges, using the Excel range syntax.

See Data Model for more information on how the adapter detects tables and how to query them.

Connecting to SharePoint Online

Set SharePointEdition to "SharePoint Online" and set the User and Password for an Azure Active Directory account.

Set the Url to a site collection to query workbooks in all nested subsites. Set the Url to a site to query workbooks in that site only.

URL	Example URL
Site	https://teams.contoso.com/teamA or https://teamA.contoso.com
Site Collection	https://teams.contoso.com

Connecting to SharePoint On Premises

Set SharePointEdition to "SharePoint OnPremise" and set the Url to your server's name or IP address. Additionally, set SharePointVersion and the authentication values.

To authenticate to SharePoint OnPremise, set AuthScheme to the authentication type and set User and Password, if necessary. *Note*: When connecting to SharePoint On-Premises 2010, you must set UseRESTAPI to true.

Windows (NTLM)

This is the most common authentication type. As such, the adapter is preconfigured to use NTLM as the default; simply set the Windows User and Password to connect.

Kerberos and Kerberos Delegation

To authenticate with Kerberos, set AuthScheme to NEGOTIATE. If needed, provide the User and Password properties. To use Kerberos Delegation, set AuthScheme to KERBEROSDELEGATION.

KerberosKDC, KerberosSPN, and KerberosRealm enable control over the components of Kerberos authentication.

Forms

This allows authentication through a custom authentication method, instead of Active Directory. To use this authentication type, set AuthScheme to FORMS and set the User and Password.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Scheme	The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, FORMS, NONE, NEGOTIATE, and KERBEROSDELEGATION.
Define Tables	Define the tables within a spreadsheet.
File	The name of the Excel file to which to connect.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
Folder	The folder containing the workbook specified by the File property.
Header	Indicates whether the first row should be used as a column header when using the REST API.

Library	The Document Library to which to connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password of the SharePoint Online account used to authenticate to the SharePoint Online server.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Row Scan Depth	The maximum number of rows to scan to look for columns available in the table. If TypeDetectionScheme is set to RowScan, this property will also control how many rows to scan in order to determine data type.
Share Point Edition	The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.
Share Point Version	The version of the SharePoint server to which you are connecting.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.

SSO Domain	The domain of the user when using Single Sign On (SSO).
STSURL	The URL of the security token service (STS) when using Single Sign On (SSO).
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Type Detection Scheme	Determines how to determine the data type of columns.
URL	The base URL for a site or site collection.
User	The username of the SharePoint Online account used to authenticate to the server.
Use RESTAPI	Whether or not the REST API is used for retrieving data.
Use SSO	Whether or not to use Single Sign On (SSO) to authenticate to SharePoint Online.

Auth Scheme

The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, FORMS, NONE, NEGOTIATE, and KERBEROSDELEGATION.

Data Type

string

Default Value

"NTLM"

Remarks

Together with Password and User, this field is used to authenticate against the server. NTLM is the default option. Use the following options to select your authentication scheme:

NTLM: Set this to use your Windows credentials for authentication.

BASIC: Set this to use HTTP Basic authentication.

DIGEST: Set this to HTTP Digest authentication.

FORMS: Set this to use Forms authentication.

NEGOTIATE: If

AuthScheme is set to NEGOTIATE, the adapter will negotiate an authentication mechanism with the server. Set AuthScheme to NEGOTIATE if you want to use Kerberos authentication.

KERBEROSDELEGATION: Set this to use delegation through the Kerberos protocol. Set the

User and Password of the account you want to impersonate.

Define Tables

Define the tables within a spreadsheet.

Data Type

string

Default Value

""

Remarks

This property is used to define the ranges within a spreadsheet that will appear as tables, when using the REST API. The value is a comma-separated list of name-value pairs in the form [Table Name]=[Sheet Name]![Range]. Table Name is the name of the table you want to use for the data and will be used when issuing queries. Sheet Name is the name of the sheet within the spreadsheet and Range is the range of cells that contain the data for the table.

Here is an example DefineTables value:
DefineTables="DefinedTable1=Sheet1!A1:N25,DefinedTable2=Sheet2!C3:M53"

Note: If the name of a defined table is the same as one returned by default (e.g. same name as a worksheet), the defined table will replace the default table.

File

The name of the Excel file to which to connect.

Data Type

string

Default Value

""

Remarks

The name of Excel file to which to connect (including the extension). The file must exist.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description

TUNNEL	80	When this is set, the adapter opens a connection to SharePoint Excel Services and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Folder

The folder containing the workbook specified by the File property.

Data Type

string

Default Value

""

Remarks

The full, hierarchical path of the subfolder in a Library where the File can be found. For example if the File is located in a folder called "SubFolder" within the folder called "BaseFolder", the property will be set to "/BaseFolder/SubFolder/".

Header

Indicates whether the first row should be used as a column header when using the REST API.

Data Type

bool

Default Value

true

Remarks

If true, the first row will be used as a column header. Otherwise, the pseudo column names A, B, C, etc. will be used.

Note: This property is only used when UseRESTAPI is 'True'.

Library

The Document Library to which to connect.

Data Type

string

Default Value

""

Remarks

This property indicates the name of the Document Library containing the Excel file. If no library is specified, the "Shared Documents" library will be used.

If you wish to connect to OneDrive for Business, set this property to "Documents".

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password of the SharePoint Online account used to authenticate to the SharePoint Online server.

Data Type

string

Default Value

""

Remarks

Together with User, this field is used to authenticate against the SharePoint Online server;.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.

TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Row Scan Depth

The maximum number of rows to scan to look for columns available in the table. If TypeDetectionScheme is set to RowScan, this property will also control how many rows to scan in order to determine data type.

Data Type

string

Default Value

"50"

Remarks

Since Excel Services is schemaless, the columns in a table must be determined by scanning table rows. This value determines the maximum number of rows that will be scanned.

Share Point Edition

The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.

Data Type

string

Default Value

"SharePoint Online"

Remarks

The edition of SharePoint being used. Set either SharePoint Online or SharePoint On-Premise.

Share Point Version

The version of the SharePoint server to which you are connecting.

Data Type

string

Default Value

"SharePoint 2013"

Remarks

Accepted entries are SharePoint 2013 and SharePoint 2010.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801c bb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

SSO Domain

The domain of the user when using Single Sign On (SSO).

Data Type

string

Default Value

""

Remarks

This property is only applicable when using Single Sign On (UseSSO is set to true) and if the domain of the User (e.g. user@mydomain.com) is different than the domain configured within the SSO service (e.g. user@myssodomain.com).

This property may be required when using OneLogin or OKTA SSO services.

STSURL

The URL of the security token service (STS) when using Single Sign On (SSO).

Data Type

string

Default Value

""

Remarks

This property only needs to be set when using Single Sign On with a local Active Directory Federation Services (ADFS).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Type Detection Scheme

Determines how to determine the data type of columns.

Data Type

string

Default Value

"RowScan"

Remarks

None	Setting TypeDetectionScheme to None will return all columns as the string type.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned.

URL

The base URL for a site or site collection.

Data Type

string

Default Value

""

Remarks

The following are examples of valid URLs:

http://server/SharePoint/

http://server/Sites/mysite/

http://server:90/

The provider will use Url to derive URLs for other calls to the server.

User

The username of the SharePoint Online account used to authenticate to the server.

Data Type

string

Default Value

""

Remarks

Together with Password, this field is used to authenticate to the SharePoint Online server specified in Url.

Use RESTAPI

Whether or not the REST API is used for retrieving data.

Data Type

string

Default Value

"False"

Remarks

Whether or not the REST API is used for retrieving data. Enable this option to use the REST API when connecting to SharePoint 2013, SharePoint 2016, and SharePoint Online. In these SharePoint versions, both the REST API and OData API are available.

Use the REST API to access spreadsheets and ranges as tables as well as table objects. The OData API enables access to only table objects.

See Data Model for more information on querying spreadsheets organized in different ways.

Use SSO

Whether or not to use Single Sign On (SSO) to authenticate to SharePoint Online.

Data Type

bool

Default Value

false

Remarks

When set to true, Single Sign On (SSO) will be used to authenticate to SharePoint Online using the account specified via User and Password. Active Directory Federation Services (ADFS), OneLogin, and OKTA SSO identity providers are supported.

SSODomain may be required to be set if the domain configured on the SSO domain is different than the domain of the User.

If you need to use a local ADFS instance to perform authentication, STSURL must be set to the URL of the local instance.

SSO is only applicable when using SharePoint Online. SSO is not supported for On-Premise versions of SharePoint.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the SharePoint Excel Services Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Advanced Settings

Fine Tuning Data Access

To make it easier to access data in advanced integrations, you can use the following:

TypeDetectionScheme: You can use this property to enable or disable automatic type detection based on the value specified in RowScanDepth.

RowScanDepth: This property determines the number of rows that will be scanned to determine column data types.

Customizing the SSL Configuration

By default, the adapter attempts to negotiate SSL/TLS by checking the server's certificate against the system's trusted certificate store. To specify another certificate, see the SSLServerCert property for the available formats to do so.

Connecting Through a Firewall or Proxy

To connect through the Windows system proxy, set only the SharePoint authentication properties and the Url. To connect to other proxies, set ProxyAutoDetect to false and in addition set the following.

To authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

To connect to other proxies, set FirewallType, FirewallServer, and FirewallPort. To tunnel the connection, set FirewallType to TUNNEL. To authenticate to a SOCKS proxy, set FirewallType to SOCKS5. Additionally, specify FirewallUser and FirewallPassword.

Troubleshooting the Connection

To show adapter activity from query execution to HTTP calls, use Logfile and Verbosity. The examples of common connection errors below show how to use these properties to get more context. Contact the support team for help tracing the source of an error or circumventing a performance issue.

Authentication errors: Typically, recording a Logfile at Verbosity 4 is necessary to get full details on an authentication error.

Queries time out: A server that takes too long to respond will exceed the adapter's client-side timeout. Often, setting the Timeout property to a higher value will avoid a connection error. Another option is to disable the timeout by setting the property to 0. Setting Verbosity to 2 will show where the time is being spent.

The certificate presented by the server cannot be validated: This error indicates that the adapter cannot validate the server's certificate through the chain of trust. (If you are using a self-signed certificate, there is only one certificate in the chain).

To resolve this error, you must verify yourself that the certificate can be trusted and specify to the adapter that you trust the certificate. One way you can specify that you trust a certificate is to add the certificate to the trusted system store; another is to set SSLServerCert.

SQL Compliance

The SharePoint Excel Services Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the SharePoint Excel Services API.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the SharePoint Excel Services adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
```

```
  [ WHERE <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
    [
      { OFFSET | , }
      <expression>
    ]
  ]
}

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | {  <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | LIKE | AND | OR } [
<expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Account
```

Rename a column:

```
SELECT "Name" AS MY_Name FROM Account
```

Search data:

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

The SharePoint Excel Services APIs support the following operators in the WHERE clause: =, >, <, >=, <=, LIKE, AND, OR.

```
SELECT * FROM Account WHERE Industry = 'Floppy Disks';
```

Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM Account
```

Sort a result set in ascending order:

```
SELECT Id, Name FROM Account  ORDER BY Name ASC
```

Predicate Functions

CEILING(value)

Returns the value rounded up to the nearest whole number (no decimal component).

expression: The value to round.

CONCAT(string_expr1, string_expr2)

Returns the string that is the concatenation of string_expr1 and string_expr2.

string_expr1: The first string to be concatenated.

string_expr2: The second string to be concatenated.

DAY(datetime_date)

Returns the integer that specifies the day component of the specified date.

datetime_date: The datetime string that specifies the date.

ENDSWITH(string_expression, string_suffix)

Returns true if string_expression ends with string_suffix, otherwise returns false.

string_expression: The string expression to search within.

string_suffix: The string suffix to search for.

FLOOR(value)

Returns the value rounded down to the nearest whole number (no decimal component).

value: The value to round.

HOUR(datetime_time)

Returns the integer that specifies the hour component of the specified time.

datetime_time: The datetime string that specifies the time.

INDEXOF(string_expression, string_search)

Returns the index location where string_search is contained within string_expression.

string_expression: The string expression to search within.

string_search: The search value to locate within string_expression.

LENGTH(string_expression)

Returns the number of characters of the specified string expression.

string_expression: The string expression.

MINUTE(datetime_time)

Returns the integer that specifies the minute component of the specified time.

datetime_time: The datetime string that specifies the time.

MONTH(datetime_date)

Returns the integer that specifies the month component of the specified date.

datetime_date: The datetime string that specifies the date.

REPLACE(string_expression, string_search, string_replace)

Returns the string after replacing any found string_search values with string_replace.

string_expression: The string expression to perform a replace on.

string_search: The string value to find within string_expression.

string_replace: The string value replace and string_search instances found.

ROUND(value)

Returns the value to the nearest whole number (no decimal component).

value: The value to round.

SECOND(datetime_time)

Returns the integer that specifies the second component of the specified time.

datetime_time: The datetime string that specifies the time.

STARTSWITH(string_expression, string_prefix)

Returns true if string_expression starts with string_prefix, otherwise returns false.

string_expression: The string expression to search within.

string_prefix: The string prefix to search for.

SUBSTRINGOF(string_expression, string_search)

Returns true if string_expression contains string_expression, otherwise returns false.

string_expression: The string expression to search within.

string_search: The value to search for.

TOLOWER(string_expression)

Returns the string_expression with the uppercase character data converted to lowercase.

string_expression: The string expression to lowercase.

TOUPPER(string_expression)

Returns the string_expression with the lowercase character data converted to uppercase.

string_expression: The string expression to uppercase.

TRIM(string_expression)

Returns the string_expression with the leading and trailing whitespace removed.

string_expression: The string expression to trim.

YEAR(datetime_date)

Returns the integer that specifies the year component of the specified date.

datetime_date: The datetime string that specifies the date.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT Id, Name INTO "csv://Account.txt" FROM "Account" WHERE
Industry = 'Floppy Disks'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT Id, Name INTO "csv://Account.txt;delimiter=tab" FROM
"Account" WHERE Industry = 'Floppy Disks'
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV SparkSQL Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

Set the Server, Database, User, and Password connection properties to connect to SparkSQL.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Auth Scheme	The authentication scheme used. Accepted entries are PLAIN, LDAP, NONE, and KERBEROS.
Database	The name of the SparkSQL database.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.
HTTP Path	The path component of the URL endpoint when using HTTP TransportMode.

Kerberos KDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
Kerberos Realm	The Kerberos Realm used to authenticate the user with.
Kerberos SPN	The Service Principal Name for the Kerberos Domain Controller.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	The password used to authenticate with SparkSQL.
Port	The port for the SparkSQL database.
Protocol Version	The Protocol Version used to authenticate with SparkSQL.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.
Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Query Passthrough	This option passes the query to SparkSQL as-is.
Readonly	You can use this property to enforce read-only access to SparkSQL from the provider.
Server	The host name or IP address of the server hosting the SparkSQL database.

Server Configurations	A name-value list of server configuration variables to override the server defaults.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Transport Mode	The transport mode to use to communicate with the Hive server. Accepted entries are BINARY and HTTP.
Use Insert Select Syntax	Specifies whether to use SSL Encryption when connecting to Hive.
User	The username used to authenticate with SparkSQL.
Use SSL	Specifies whether to use SSL Encryption when connecting to Hive.

Auth Scheme

The authentication scheme used. Accepted entries are PLAIN, LDAP, NONE, and KERBEROS.

Data Type

string

Default Value

"PLAIN"

Remarks

The AuthScheme used to authenticate with SparkSQL.

Database

The name of the SparkSQL database.

Data Type

string

Default Value

""

Remarks

The name of the SparkSQL database.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description

TUNNEL	80	When this is set, the adapter opens a connection to SparkSQL and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

HTTP Path

The path component of the URL endpoint when using HTTP TransportMode.

Data Type

string

Default Value

"cliservice"

Remarks

This property is used to specify the path component of the URL endpoint when using HTTP TransportMode.

This property should be set to the value specified in the 'hive.server2.thrift.http.path' property of you Hive configuration file (hive-site.xml).

Kerberos KDC

The Kerberos Key Distribution Center (KDC) service used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos Key Distribution Center (KDC) service. The Kerberos Key Distribution Center (KDC) service is conventionally colocated with the domain controller. If Kerberos KDC is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using the config file krb5.conf, or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm

and KerberosKDC are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the configured domain name and host as a last resort.

Note: Windows authentication is supported in JRE 1.6 and above only.

Kerberos Realm

The Kerberos Realm used to authenticate the user with.

Data Type

string

Default Value

""

Remarks

The Kerberos properties are used when using SPNEGO or Windows Authentication. The Kerberos Realm is used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name. If Kerberos Realm is not specified the adapter will attempt to detect these properties automatically from the following locations:

Java System Properties: Kerberos settings can be configured in Java using a config file (krb5.conf) or using the system properties java.security.krb5.realm and java.security.krb5.kdc. The adapter will use the system settings if KerberosRealm and KerberosKDC

are not explicitly set.

Domain Name and Host: The adapter will infer the Kerberos Realm and Kerberos KDC from the user-configured domain name and host as a last resort. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

Kerberos SPN

The Service Principal Name for the Kerberos Domain Controller.

Data Type

string

Default Value

""

Remarks

If the Service Principal Name on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, set the Service Principal Name here.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

The password used to authenticate with SparkSQL.

Data Type

string

Default Value

""

Remarks

The password used to authenticate with SparkSQL.

Port

The port for the SparkSQL database.

Data Type

string

Default Value

"27017"

Remarks

The port for the SparkSQL database.

Protocol Version

The Protocol Version used to authenticate with SparkSQL.

Data Type

string

Default Value

"8"

Remarks

The Protocol Version used to authenticate with SparkSQL.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Query Passthrough

This option passes the query to SparkSQL as-is.

Data Type

bool

Default Value

false

Remarks

The Protocol Version used to authenticate with SparkSQL.

Readonly

You can use this property to enforce read-only access to SparkSQL from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Server

The host name or IP address of the server hosting the SparkSQL database.

Data Type

string

Default Value

""

Remarks

The host name or IP address of the server hosting the SparkSQL database.

Server Configurations

A name-value list of server configuration variables to override the server defaults.

Data Type

string

Default Value

""

Remarks

This property takes a comma separated list of configuration variables specified as name-value pairs. Any values specified here will be sent to the Hive server to override the default values.

Example: hive.enforce.bucketing=true,hive.enforce.sorting=true

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.....Q w== -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Transport Mode

The transport mode to use to communicate with the Hive server. Accepted entries are BINARY and HTTP.

Data Type

string

Default Value

"BINARY"

Remarks

The transport mode used to communicate with Hive server.

This property should be set to the 'hive.server2.transport.mode' value specified in your Hive configuration file (hive-site.xml).

Use Insert Select Syntax

Specifies whether to use SSL Encryption when connecting to Hive.

Data Type

bool

Default Value

false

Remarks

Set this property to the value specified in the 'hive.server2.use.SLL' property of your Hive configuration file (hive-site.xml).

User

The username used to authenticate with SparkSQL.

Data Type

string

Default Value

""

Remarks

The username used to authenticate with SparkSQL.

Use SSL

Specifies whether to use SSL Encryption when connecting to Hive.

Data Type

bool

Default Value

false

Remarks

Set this property to the value specified in the 'hive.server2.use.SLL' property of your Hive configuration file (hive-site.xml).

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the SparkSQL Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

SQL Compliance

The SparkSQL Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the SparkSQL API.

INSERT Statements

See INSERT Statements for a syntax reference and examples.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately. Or, use Batch Processing.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the SparkSQL adapter:

```
SELECT {
  [ TOP <numeric_literal> | DISTINCT ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  } [ , ... ]
  [ [
      INNER | { { LEFT | RIGHT | FULL  } [ OUTER ] }
    ] JOIN <table_reference> [ ON <search_condition> ] [ [ AS ]
<identifier> ]
  ] [ ... ]
  [ WHERE <search_condition> ]
  [ GROUP BY <column_reference> [ , ... ]
  [ HAVING <search_condition> ]
  [
    ORDER BY
    { <column_reference> [ ASC | DESC ] } [ , ... ]
  ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()

  <expression> ::=
    | <column_reference>
    | @ <parameter>
    | ?
    | COUNT( * | { [ DISTINCT ] <expression> } )
    | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
    | <literal>
    | <sql_function>

  <search_condition> ::=
    {
      <expression> { = | > | < | >= | <= | <> | != | LIKE | IN | NOT
IN | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]
```

Examples

Return all columns:
```
SELECT * FROM Customers
```

Rename a column:
```
SELECT "CompanyName" AS MY_CompanyName FROM Customers
```

Search data:
```
SELECT * FROM Customers WHERE Country = 'US';
```

The SparkSQL APIs support the following operators in the WHERE clause: =, >, <, >=, <=, <>, !=, LIKE, IN, NOT IN, AND, OR.
```
SELECT * FROM Customers WHERE Country = 'US';
```

Return the number of items matching the query criteria:
```
SELECT COUNT(*) AS MyCount FROM Customers
```

Return the number of unique items matching the query criteria:
```
SELECT COUNT(DISTINCT CompanyName) FROM Customers
```

Return the unique items matching the query criteria:
```
SELECT DISTINCT CompanyName FROM Customers
```

Summarize data:
```
SELECT CompanyName, MAX(Balance) FROM Customers  GROUP BY
CompanyName
```

See Aggregate Functions for details.

Retrieve data from multiple tables.
```
SELECT "restaurants"."restaurant_id", "restaurants".name,
"restaurants.grades".* FROM "restaurants.grades" JOIN
"restaurants" WHERE "restaurants".name = 'Morris Park Bake Shop'
```

See JOIN Queries for details.

Sort a result set in ascending order:
```
SELECT City, CompanyName FROM Customers  ORDER BY CompanyName ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM Customers WHERE Country = 'US'
```

COUNT_DISTINCT

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT_DISTINCT(City) AS DistinctValues FROM Customers WHERE
Country = 'US'
```

COUNT(DISTINCT)

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT(DISTINCT City) AS DistinctValues FROM Customers WHERE
Country = 'US'
```

AVG

Returns the average of the column values.

```
SELECT CompanyName, AVG(Balance) FROM Customers WHERE Country =
'US' GROUP BY CompanyName
```

MIN

Returns the minimum column value.

```
SELECT MIN(Balance), CompanyName FROM Customers WHERE Country =
'US' GROUP BY CompanyName
```

MAX

Returns the maximum column value.

```
SELECT CompanyName, MAX(Balance) FROM Customers WHERE Country =
'US' GROUP BY CompanyName
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(Balance) FROM Customers WHERE Country = 'US'
```

JOIN Queries

The SparkSQL Adapter supports joins of a nested array with its parent document
.

Projection Functions

JSON_AVG(json, jsonpath)

Computes the average value of a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_COUNT(json, jsonpath)

Returns the number of elements in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MAX(json, jsonpath)

Gets the maximum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_MIN(json, jsonpath)

Gets the minimum value in a JSON array within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_SUM(json, jsonpath)

Computes the sum of the elements in a JSON within a JSON object.

json: The column containing JSON data.

jsonpath: The path to the json array.

JSON_EXTRACT(json, jsonpath)

Selects any value in a JSON array or object. The path to the array is specified in the jsonpath argument. Return value is numeric or null.

json: The JSON document to extract.

jsonpath: The XPath used to select the nodes. The JSONPath must be a string constant. The values of the nodes selected will be returned in a token-separated list.

XML_EXTRACT(xml, xpath [, separator])

Extracts an XML document using the specified XPath to flatten the XML. A comma is used to separate the outputs by default, but this can be changed by specifying the third parameter.

xml: The XML document to extract.

xpath: The XPath used to select the nodes. The nodes selected will be returned in a token-separated list.

separator: The optional token used to separate the items in the flattened response. If this is not specified, the separator will be a comma.

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT City, CompanyName INTO "csv://Customers.txt" FROM
"Customers" WHERE Country = 'US'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT City, CompanyName INTO "csv://Customers.txt;delimiter=tab"
FROM "Customers" WHERE Country = 'US'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO Customers (CompanyName) VALUES (?)";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "RSSBus Inc.");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
```

```
| ?
| <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

Data Model

A full list is available upon customer request.

TDV Twitter Adapter

Requirements and Restrictions

The Supported SQL section shows the SQL syntax supported by the adapter and points out any limitations.

Basic Tab

All tables require authentication. You can connect using your User and Password or OAuth. OAuth requires the authenticating user to interact with Twitter using the browser. For more information, refer to our Connecting to Twitter guide.

Advanced Tab

The connection string properties describe the various options that can be used to establish a connection.

Connection String Options

The following is the full list of the options you can configure in the connection string for this provider.

Account Id	Identifier of the advertising account. This Id is used to get analytic stats for the account that is identified by this Id. The AddAccounts view can be used to list available accounts with their specific Ids.
Callback URL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
Firewall Password	A password used to authenticate to a proxy-based firewall.
Firewall Port	The TCP port for a proxy-based firewall.
Firewall Server	The name or IP address of a proxy-based firewall.
Firewall Type	The protocol used by a proxy-based firewall.
Firewall User	The user name to use to authenticate with a proxy-based firewall.

Initiate OAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.
Max Rate Limit Delay	The maximum amount of time to delay (in seconds) before submitting a request if it would be rate limited.
OAuth Access Token	The access token for connecting using OAuth.
OAuth Access Token Secret	The OAuth access token secret for connecting using OAuth.
OAuth Client Id	The client Id assigned when you register your application with an OAuth authorization server.
OAuth Client Secret	The client secret assigned when you register your application with an OAuth authorization server.
OAuth Settings Location	The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.
Other	The other parameters necessary to connect to a data source, such as username and password, when applicable.
Password	Your Twitter password. Specify only if the OAuth access token is not specified.
Proxy Auth Scheme	The authentication type to use to authenticate to the ProxyServer proxy.
Proxy Auto Detect	This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.
Proxy Exceptions	A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .
Proxy Password	A password to be used to authenticate to the ProxyServer proxy.
Proxy Port	The TCP port the ProxyServer proxy is running on.
Proxy Server	The hostname or IP address of a proxy to route HTTP traffic through.

Proxy SSL Type	The SSL type to use when connecting to the ProxyServer proxy.
Proxy User	A user name to be used to authenticate to the ProxyServer proxy.
Readonly	You can use this property to enforce read-only access to Twitter from the provider.
Search Terms	Default SearchTerms if none are specified. Used for some tables such as Tweets where SearchTerms may be specified.
SSL Server Cert	The certificate to be accepted from the server when connecting using TLS/SSL.
Stream Page Size	The number of results to return per page of data retrieved from the Twitter stream.
Stream Timeout	The maximum number of seconds to continue waiting for results while streaming. When this value is reached and no tweets are returned, then the connection will be closed.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
Use App Only Authentication	A boolean that indicates whether or not you would like to use app-only authentication.
User	Your Twitter user account. Specify only if the access token is not specified.

Account Id

Identifier of the advertising account. This Id is used to get analytic stats for the account that is identified by this Id. The AddAccounts view can be used to list available accounts with their specific Ids.

Data Type

string

Default Value

""

Remarks

Identifier of the advertising account. This Id is used to get analytic stats for the account that is identified by this Id. The AddAccounts view can be used to list available accounts with their specific Ids.

Callback URL

The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.

Data Type

string

Default Value

""

Remarks

During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

Firewall Password

A password used to authenticate to a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property is passed to the proxy specified by FirewallServer and FirewallPort, following the authentication method specified by FirewallType.

Firewall Port

The TCP port for a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This specifies the TCP port for a proxy allowing traversal of a firewall. Use FirewallServer to specify the name or IP address. Specify the protocol with FirewallType.

Firewall Server

The name or IP address of a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

This property specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by FirewallType: Use FirewallServer with this property to connect through SOCKS or do tunneling. Use ProxyServer to connect to an HTTP proxy.

Note that the adapter uses the system proxy by default. To use a different proxy, set ProxyAutoDetect to false.

Firewall Type

The protocol used by a proxy-based firewall.

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the FirewallServer proxy. Note that by default the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set ProxyAutoDetect to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Twitter and traffic flows back and forth through the proxy.
SOCKS4	1080	When this is set, the adapter sends data through the SOCKS 4 proxy specified by FirewallServer and FirewallPort and passes the FirewallUser value to the proxy, which determines if the connection request should be granted.
SOCKS5	1080	When this is set, the adapter sends data through the SOCKS 5 proxy specified by FirewallServer and FirewallPort. If your proxy requires authentication, set FirewallUser and FirewallPassword to credentials the proxy recognizes.

To connect to HTTP proxies, use ProxyServer and ProxyPort. To authenticate to HTTP proxies, use ProxyAuthScheme, ProxyUser, and ProxyPassword.

Firewall User

The user name to use to authenticate with a proxy-based firewall.

Data Type

string

Default Value

""

Remarks

The FirewallUser and FirewallPassword properties are used to authenticate against the proxy specified in FirewallServer and FirewallPort, following the authentication method specified in FirewallType.

Initiate OAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Data Type

string

Default Value

"OFF"

Remarks

The following options are available:

OFF: Indicates that the OAuth flow will be handled entirely by the user. An OAuthAccessToken will be required to authenticate.

GETANDREFRESH: Indicates that the entire OAuth Flow will be handled by the adapter. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.

REFRESH: Indicates that the adapter will only handle refreshing the OAuthAccessToken. The user will never be prompted by the adapter to authenticate via the browser. The user must handle obtaining the OAuthAccessToken and OAuthRefreshToken initially.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

""

Remarks

The path to a directory which contains the schema files for the adapter (.rsd files for tables and views, .rsb files for stored procedures). The Location property is only needed if you would like to customize definitions (e.g., change a column name, ignore a column, etc.) or extend the data model with new tables, views, or stored procedures.

The schema files are deployed alongside the adapter assemblies. You must also ensure that Location points to the folder that contains the schema files. The folder location can be a relative path from the location of the executable.

Max Rate Limit Delay

The maximum amount of time to delay (in seconds) before submitting a request if it would be rate limited.

Data Type

string

Default Value

"60"

Remarks

Twitter uses different rate limits for total number of requests for different endpoints. These can range from as few as 15 per 15 minute window, up to 900 for a 15 minute window. Internally the Twitter Adapter keeps track of if a given request would result in a rate limit exception. If a rate limit would occur, the Twitter Adapter can internally delay submitting a request until the limit is up. However, this could also result in waiting for several minutes before requesting data, which is also not a good behavior.

The MaxRateLimitDelay gives control over the maximum amount of time the Twitter Adapter will wait once it detects a rate limit would occur. Since the amount of time the Twitter Adapter needs to wait can be calculated, if it would have to wait longer than the MaxRateLimitDelay, it will simply error immediately when it sees the time would take too long.

OAuth Access Token

The access token for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessToken property is used to connect using OAuth. The OAuthAccessToken is retrieved from the OAuth server as part of the authentication process. It has a server-dependent timeout and can be reused between requests.

The access token is used in place of your username and password. The access token protects your credentials by keeping them on the server.

OAuth Access Token Secret

The OAuth access token secret for connecting using OAuth.

Data Type

string

Default Value

""

Remarks

The OAuthAccessTokenSecret property is used to connect and authenticate using OAuth. The OAuthAccessTokenSecret is retrieved from the OAuth server as part of the authentication process. It is used with the OAuthAccessToken and can be used for multiple requests until it times out.

OAuth Client Id

The client Id assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId value, sometimes also called a consumer key, and a client secret, the OAuthClientSecret.

OAuth Client Secret

The client secret assigned when you register your application with an OAuth authorization server.

Data Type

string

Default Value

""

Remarks

As part of registering an OAuth application, you will receive the OAuthClientId, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the OAuthClientSecret property.

OAuth Settings Location

The location of the settings file where OAuth values are saved when InitiateOAuth is set to GETANDREFRESH or REFRESH.

Data Type

string

Default Value

"%APPDATA%\\CData\\Twitter Data Provider\\OAuthSettings.txt"

Remarks

When InitiateOAuth is set to GETANDREFRESH or REFRESH, the adapter saves OAuth values to a settings file to avoid requiring the user to manually enter OAuth connection properties. The default OAuthSettingsLocation is a settings file located in the %AppData%\CData folder.

Other

The other parameters necessary to connect to a data source, such as username and password, when applicable.

Data Type

string

Default Value

""

Remarks

The Other property is a semicolon-separated list of name-value pairs used in connection parameters specific to a data source.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Password

Your Twitter password. Specify only if the OAuth access token is not specified.

Data Type

string

Default Value

""

Remarks

A password that can be used to authenticate to Twitter. Performs an XAuth authentication that does not require an access token or secret key, but requires an OAuthClientId and OAuthClientSecret. Please note that XAuth provides more limited access to Twitter information and does not allow access to direct messages.

Proxy Auth Scheme

The authentication type to use to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

"BASIC"

Remarks

This value specifies the authentication type to use to authenticate to the HTTP proxy specified by ProxyServer and ProxyPort.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to connect to another proxy, you will need to set ProxyAutoDetect to false, in addition to ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

The authentication type can be one of the following:

BASIC: The adapter performs HTTP BASIC authentication.

DIGEST: The adapter performs HTTP DIGEST authentication.

NEGOTIATE: The adapter retrieves an NTLM or Kerberos token based on the applicable protocol for authentication.

PROPRIETARY: The adapter does not generate an NTLM or Kerberos token. You must supply this token in the Authorization header of the HTTP request.

If you need to use another authentication type, such as SOCKS 5 authentication, see FirewallType.

Proxy Auto Detect

This indicates whether to use the system proxy settings or not. Set ProxyAutoDetect to FALSE to use custom proxy settings. This takes precedence over other proxy settings.

Data Type

bool

Default Value

true

Remarks

By default, the adapter uses the system HTTP proxy. Set this to false if you want to connect to another proxy.

To connect to an HTTP proxy, see ProxyServer.

For other proxies, such as SOCKS or tunneling, see FirewallType.

Proxy Exceptions

A semicolon separated list of hosts or IPs that will be exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The ProxyServer will be used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter will use the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you will need to set ProxyAutoDetect to false, and configure ProxyServer and ProxyPort. To authenticate, set ProxyAuthScheme and set ProxyUser and ProxyPassword, if needed.

Proxy Password

A password to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set ProxyServer and ProxyPort. To specify the authentication type, set ProxyAuthScheme.

If you are using HTTP authentication, additionally set ProxyUser and ProxyPassword to HTTP proxy.

If you are using NTLM authentication, set ProxyUser and ProxyPassword to your Windows password. You may also need these to complete Kerberos authentication.

For SOCKS 5 authentication or tunneling, see FirewallType.

By default, the adapter uses the system proxy. If you want to connect to another proxy, set ProxyAutoDetect to false.

Proxy Port

The TCP port the ProxyServer proxy is running on.

Data Type

string

Default Value

"80"

Remarks

The port the HTTP proxy is running on that you want to redirect HTTP traffic through. Specify the HTTP proxy in ProxyServer. For other proxy types, see FirewallType.

Proxy Server

The hostname or IP address of a proxy to route HTTP traffic through.

Data Type

string

Default Value

""

Remarks

The hostname or IP address of a proxy to route HTTP traffic through. The adapter can use the HTTP, Windows (NTLM), or Kerberos authentication types to authenticate to an HTTP proxy.

If you need to connect through a SOCKS proxy or tunnel the connection, see FirewallType.

By default, the adapter uses the system proxy. If you need to use another proxy, set ProxyAutoDetect to false.

Proxy SSL Type

The SSL type to use when connecting to the ProxyServer proxy.

Data Type

string

Default Value

"AUTO"

Remarks

This property determines when to use SSL for the connection to an HTTP proxy specified by ProxyServer. This value can be AUTO, ALWAYS, NEVER, or TUNNEL. The applicable values are the following:

AUTO	Default setting. If the URL is an HTTPS URL, the adapter will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
ALWAYS	The connection is always SSL enabled.
NEVER	The connection is not SSL enabled.
TUNNEL	The connection is through a tunneling proxy: The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

Proxy User

A user name to be used to authenticate to the ProxyServer proxy.

Data Type

string

Default Value

""

Remarks

The ProxyUser and ProxyPassword options are used to connect and authenticate against the HTTP proxy specified in ProxyServer.

You can select one of the available authentication types in ProxyAuthScheme. If you are using HTTP authentication, set this to the username of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a username in one of the following formats:

```
user@domain
domain\user
```

Readonly

You can use this property to enforce read-only access to Twitter from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

Search Terms

Default SearchTerms if none are specified. Used for some tables such as Tweets where SearchTerms may be specified.

Data Type

string

Default Value

""

Remarks

Default SearchTerms if none are specified. Used for some tables such as Tweets where SearchTerms may be specified.

SSL Server Cert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine will be rejected.

This property can take the forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIIChTCCAe4CAQAwDQYJKoZIhv.. ....Qw== -----END CERTIFICATE-----

A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	ecadbdda5a1529c58a1e9e09828d70e4
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34a929226ae0819f2ec14b4a3d904f801cbb150d

If not specified, any certificate trusted by the machine will be accepted. Use '*' to signify to accept all certificates (not recommended for security concerns).

Stream Page Size

The number of results to return per page of data retrieved from the Twitter stream.

Data Type

string

Default Value

"50"

Remarks

The number of results to return per page of data retrieved from the Twitter stream.

Stream Timeout

The maximum number of seconds to continue waiting for results while streaming. When this value is reached and no tweets are returned, then the connection will be closed.

Data Type

string

Default Value

"0"

Remarks

Set the value of StreamTimeout to 0 in order to keep the connection open indefinitely. Note that, if the value of this property is greater than zero, the value of the StreamPageSize property, will be overwriten and will be set to one(1).

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

string

Default Value

"60"

Remarks

If the Timeout property is set to 0, operations do not time out: They run until they complete successfully or encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

Use App Only Authentication

A boolean that indicates whether or not you would like to use app-only authentication.

Data Type

bool

Default Value

false

Remarks

Set this to true to have your Twitter app log in to Twitter instead of a user.

User

Your Twitter user account. Specify only if the access token is not specified.

Data Type

string

Default Value

""

Remarks

A Twitter user account that can be used to authenticate to Twitter. Performs an XAuth authentication that does not require an access token or secret key, but requires an OAuthClientId and OAuthClientSecret. Please note that XAuth provides more limited access to Twitter information and does not allow access to direct messages.

Logging

The adapter uses log4j to generate log files. The settings within the log4j configuration file will be used by the adapter to determine the type of messages to log. The following categories can be specified:

Error: Only error messages will be logged.

Info: Both Error and Info messages will be logged.

Debug: Error, Info, and Debug messages will be logged.

The Other property of the adapter can be used to set Verbosity to specify the amount of detail to be included in the log file, i.e.

```
Verbosity=4;
```

You can use Verbosity to specify the amount of detail to include in the log within a category. The following verbosity levels are mapped to the log4j categories:

0 = Error

1-2 = Info

3-5 = Debug

For example, if the log4j category is set to DEBUG, the Verbosity option can be set to 3 for the minimum amount of debug information or 5 for the maximum amount of debug information.

Note that the log4j settings override the Verbosity level specified. The adapter will never log at a Verbosity level greater than what is configured in the log4j properties. In addition, if Verbosity is set to a level less than the log4j category configured, Verbosity will default to the minimum value for that particular category. For example, if Verbosity is set to a value less than three and the Debug category is specified, the Verbosity will default to 3.

Here is a breakdown of the Verbosity levels and the information that they log:

1 - Will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.

2 - Will log everything included in Verbosity 1 and HTTP headers.

3 - Will additionally log the body of the HTTP requests.

4 - Will additionally log transport-level communication with the data source. This includes SSL negotiation.

5 - Will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

Configure Logging for the Twitter Adapter

By default, logging is turned on without debugging. If debugging information is desired, the following line from the TDV Server's log4j.properties file can be uncommented (default location of this file is: C:\Program Files\TIBCO\TDV Server 7.0\conf\server).

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server will need to be restarted after changing the log4j.properties file, which can be accomplished by running the composite.bat script located at C:\Program Files\TIBCO\TDV Server 7.0\conf\server. Note reauthenticating to the TDV Studio will be required after restarting the server.

An example of the calls would be:

```
.\composite.bat monitor restart
```

All logs for the adapter will be written to the "cs_cdata.log" file as specified in the log4j properties.

Note the "log4j.logger.com.cdata=DEBUG" option is not required if the "Debug Output Enabled" option is set to true within the TDV Studio. To accomplish this, navigate to Administrator -> Configuration to display the configuration window. Then expand Server -> Configuration -> Debugging and set the Debug Output Enabled option to True.

Connecting to Twitter

OAuth enables the adapter to obtain limited access to the service on behalf of a user by orchestrating an approval interaction between the user and the service. This section specifies how the adapter can be configured to authenticate using OAuth.

Configuring the Adapter

The adapter can be configured for OAuth in three ways using the InitiateOAuth property.

Set InitiateOAuth to OFF

In this configuration the adapter uses the OAuthAccessToken for authorization without initiating the OAuth flow. When the OAuthAccessToken expires, the user must obtain a new OAuthAccessToken.

This configuration is suitable for one-time use or when the OAuthAccessToken has a long life. This is the simplest way of configuring OAuth if the OAuthAccessToken can be obtained using other means. For example, with the help of the API or developer console.

Set InitiateOAuth to REFRESH

The REFRESH configuration does not require user interaction. This configuration is suitable for scenarios where the studio and server are not on the same machine.

The adapter will refresh the OAuthAccessToken when it expires. The adapter will store the new OAuthAccessToken in the OAuthSettingsLocation configured in the data source.

The following connection properties need to be set:

OAuthRefreshToken

OAuthClientId

OAuthClientSecret

OAuthSettingsLocation

Set InitiateOAuth to GETANDREFRESH

This configuration requires interaction between the adapter, user, and Twitter.

The adapter will launch the browser to allow the user to log in and grant permissions. Since this configuration requires browser interaction, there are limitations on when this configuration can be used. For example, this configuration cannot be used when the TDV monitor is used to start the server.

The following connection properties need to be set:

OAuthClientId

OAuthClientSecret

CallbackURL

OAuthSettingsLocation

Configuring a Development Machine

To configure the adapter on a development machine, where the server and studio are running on the same machine, start TDV from a console and set InitiateOAuth to GETANDREFRESH. The resulting OAuthAccessToken will be saved in OAuthSettingsLocation.

Configuring a Production Machine

If the imported data source can be used on the target server without additional modifications, copy the settings file to the target server. Note that the OAuthSettingsLocation must not change.

Alternatively, a refresh token can be obtained from Twitter and can be provided along with the OAuthClientId and OAuthClientSecret. Getting a refresh token requires experience in using the developer APIs and console of the OAuth provider and the token has to be obtained using the same OAuthClientId and OAuthClientSecret configured in the data source.

If Twitter issues a long-lived access token, use the Twitter developer API or console to retrieve the OAuthAccessToken.

Configuring OAuth in a Cluster

In a cluster the data source configuration is synced across the members of the cluster. Set the OAuthSettingsLocation to a file path that is on a shared file system that is accessible to all the members.

Importing/Exporting Archives and Using the Deployment Manager

When the archive contains a data source configured for OAuth and when it is imported or migrated to the target server, the OAuthSettingsLocation is not automatically imported or migrated. The OAuthSettingsLocation needs to be externally migrated or the OAuth flow has to be reinitiated in the target server.

2017 Changes

In 2017, we have made some changes to make the Twitter Adapter more intuitive and expansive.

Ads Support

We now support many Twitter Ads requests including retrieving your ad stats and campaign information. Analytics information can be retrieved from the AdStats and AdInsights views while other entities may be retrieved from the appropriate views. The following views have been added:

AdAccounts

AdAvailableAudiences

AdCampaigns

AdFundingInstruments

AdInsights

AdLineItems

AdPromotedTweets

AdStats

Rate Limit Handling

In the 2017 version, we have made changes to account for Twitter rate limits. Now the Twitter Adapter will automatically attempt to detect if a given request would be rate limited. If so, it will check when the rate limit will expire and automatically attempt to delay submitting the next request up to a certain point. This is configurable via MaxRateLimitDelay.

Table Changes

We've modified how some tables work. Previously we took advantage of response columns to give access to making more advanced requests against Twitter for different kinds of information on the same table. We still want to offer this, but feel it was a mistake to use the same response columns as inputs with a different meaning when used in the WHERE clause. All of these special columns have been replaced with input only columns. In addition, we've removed some redundant columns and promoted some pseudo-columns to full ones. These include:

Followers - specifying User_Id and Screen_Name in the filter no longer affects the query submitted to Twitter. Instead, use Followers_Of_User_Id and Followers_Of_Screen_Name to get back users following a specific user.

Following - specifying User_Id and Screen_Name in the filter no longer affects the query submitted to Twitter. Instead, use Following_User_Id and Following_Screen_Name to get back users following a specific user.

Tweets - promoted SearchTerms to a full (input only) column. Removed User_Id and Screen_Name pseudo-columns since they are redundant. Just use From_User_Id and From_Screen_Name.

SQL Compliance

The Twitter Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

See SELECT Statements for a syntax reference and examples.

See Data Model for information on the capabilities of the Twitter API.

INSERT Statements

See INSERT Statements for a syntax reference and examples, as well as retrieving the new records' Ids.

UPDATE Statements

The primary key Id is required to update a record. See UPDATE Statements for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See DELETE Statements for a syntax reference and examples.

REPLICATE Statements

REPLICATE statements allow high-level control over caching and incremental updates. For a syntax reference and examples, see REPLICATE Statements.

You can also incrementally update a cache automatically; see AutoCache for more information.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See EXECUTE Statements for a syntax reference and examples.

Names and Quoting

Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-Za-z0-9_:@].

To use a table or column name with characters not listed above, the name must be quoted using double quotes ("name") in any SQL statement.

Strings must be quoted using single quotes (e.g., 'John Doe').

Transactions and Batching

Transactions are not currently supported.

Additionally, the adapter does not support batching of SQL statements. To execute multiple commands, you can create multiple instances and execute each separately.

SELECT Statements

A SELECT statement can consist of the following basic clauses.

SELECT

INTO

FROM

JOIN

WHERE

GROUP BY

HAVING

UNION

ORDER BY

LIMIT

SELECT Syntax

The following syntax diagram outlines the syntax supported by the Twitter adapter:

```
SELECT {
  [ TOP <numeric_literal> ]
  {
    *
    | {
        <expression> [ [ AS ] <column_reference> ]
        | { <table_name> | <correlation_name> } .*
      } [ , ... ]
  }
  [ INTO csv:// [ filename= ] <file_path> [ ;delimiter=tab ] ]
  {
    FROM <table_reference> [ [ AS ] <identifier> ]
  }
  [ WHERE <search_condition> ]
  [
    LIMIT <expression>
  ]
} | SCOPE_IDENTITY()
```

```
<expression> ::=
  | <column_reference>
  | @ <parameter>
  | ?
  | COUNT( * | {  <expression> } )
  | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
  | <literal>
  | <sql_function>

<search_condition> ::=
  {
    <expression> { = | AND } [ <expression> ]
  } [ { AND | OR } ... ]
```

Examples

Return all columns:

```
SELECT * FROM Tweets
```

Rename a column:

```
SELECT "Text" AS MY_Text FROM Tweets
```

Search data:

```
SELECT * FROM Tweets WHERE From_User_Name = 'twitter';
```

The Twitter APIs support the following operators in the WHERE clause: =, AND.

```
SELECT * FROM Tweets WHERE From_User_Name = 'twitter';
```

SELECT INTO Statements

You can use the SELECT INTO statement to export formatted data to a file.

Data Export with an SQL Query

The following query exports data into a file formatted in comma-separated values (CSV):

```
SELECT From_User_Name, Text INTO "csv://Tweets.txt" FROM "Tweets"
WHERE From_User_Name = 'twitter'
```

You can specify other formats in the file URI. The possible delimiters are tab, semicolon, and comma with the default being comma. The following example exports tab-separated values:

```
SELECT From_User_Name, Text INTO "csv://Tweets.txt;delimiter=tab"
FROM "Tweets" WHERE From_User_Name = 'twitter'
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )


<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected. To retrieve the Id of the last inserted record use getGeneratedKeys. Additionally, set the RETURN_GENERATED_KEYS flag of the Statement class when you call prepareStatement.

```
String cmd = "INSERT INTO Tweets (Text) VALUES (?)";
PreparedStatement pstmt =
connection.prepareStatement(cmd,Statement.RETURN_GENERATED_KEYS);
pstmt.setString(1, "My twitter message 2");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
ResultSet rs = pstmt.getGeneratedKeys();
while(rs.next()){
  System.out.println(rs.getString("Id"));
}
connection.close();
```

DELETE Statements

To delete from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause.

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows.

```
Connection connection =
DriverManager.getConnection("jdbc:twitter:InitiateOAuth=GETANDREFR
ESH;",);
String cmd = "DELETE FROM Tweets WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "123456789");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements. EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]

<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

EXECUTE my_proc second = @p1, first = @p2, third = @p3;

Data Model

A full list is available upon customer request.