



TIBCO® Data Virtualization

OData Adapter Guide

Version 8.7.0 | October 2023

Contents

Contents	2
OData Adapter	4
Getting Started	4
Basic Tab	5
Logging	7
Using Kerberos	8
Using OAuth Authentication	10
Fine-Tuning Data Access	15
Changelog	17
Advanced Features	22
User Defined Views	23
Inserting Parent and Child Records	25
SSL Configuration	34
Firewall and Proxy	34
Query Processing	35
Logging	36
SQL Compliance	39
SELECT Statements	40
Predicate Functions	42
SELECT INTO Statements	47
INSERT Statements	48
UPDATE Statements	49
DELETE Statements	49
EXECUTE Statements	50
PIVOT and UNPIVOT	51
Data Model	52
Tables	53

Views	57
Stored Procedures	58
Data Type Mapping	68
Connection String Options	69
Authentication	76
Azure Authentication	80
SSO	83
OAuth	83
Kerberos	96
SSL	101
Firewall	107
Proxy	111
Logging	117
Schema	118
Miscellaneous	119
TIBCO Product Documentation and Support Services	136
How to Access TIBCO Documentation	136
How to Contact TIBCO Support	137
Release Version Support	137
How to Join TIBCO Community	138
Legal and Third-Party Notices	139

OData Adapter

OData Version Support

The adapter is a standard OData consumer that can read and write to OData 2.0, 3.0, and 4.0 services. The major authentication schemes are supported, including HTTP Basic, Digest, and NTLM, as well as SSL/TLS. The adapter also facilitates connecting to data sources that use the OAuth authentication standard.

SQL Compliance

The [SQL Compliance](#) section shows the SQL syntax supported by the adapter and points out any limitations.

Getting Started

Connecting to OData

[Basic Tab](#) shows how to authenticate to OData and configure any necessary connection properties. Additional adapter capabilities can be configured using the available [Connection](#) properties on the Advanced tab. The Advanced Settings section shows how to set up more advanced configurations and troubleshoot connection errors.

Deploying the OData Adapter

To deploy the adapter, you can execute the server_util utility via the command line by

1. Unzip the tdv.odata.zip file to the location of your choice.
2. Open a command prompt window.
3. Navigate to the <TDV_install_dir>/bin
4. Enter the server_util command with the -deploy option:

```
server_util -server <hostname> [-port <port>] -user <user> -  
password <password> -deploy -package <TDV_install_
```

```
dir>/adapters/tdv.odata/tdv.odata.jar
```

Note: When deploying a build of an existing adapter, you will need to undeploy the existing adapter using the `server_util` command with the `-undeploy` option.

```
server_util -server <hostname> [-port <port>] -user <user> -password  
<password> -undeploy -version 1 -name OData
```

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.

Authenticating to OData

The adapter supports following authentication schemes.

HTTP Auth Schemes

The following general HTTP Auth schemes are supported:

- **None:** If no authentication is required, set the AuthScheme to None.
- **Basic:** If Basic Auth is supported, set the AuthScheme to Basic. In addition, set the User and Password.
- **NTLM:** A type of Windows authentication often used across a LAN using your Windows user credentials. Set the AuthScheme to NTLM to support this method of authentication. In addition, set the User and Password if you are not connecting from a Windows machine, or your currently logged in user account should not be used for the connection.
- **Digest:** If Digest Auth is supported, set the AuthScheme to Digest. In addition, set the User and Password.

Kerberos

Please see [Using Kerberos](#) for a description of how to support Kerberos authentication.

OAuth

Set the AuthScheme to **OAuth**. See [Using OAuth Authentication](#) for an authentication guide.

Authenticating with AzureAD

AzureAD is a form of OAuth that goes through Azure. Set the AuthScheme to **AzureAD**. The OData Adapter will internally automatically take care of known Azure URLs. Specifically, the following are unnecessary to specify with the AzureAD AuthScheme:

- OAuthAccessTokenURL
- OAuthAuthorizationURL
- OAuthRefreshTokenURL
- OAuthRequestTokenURL

Other connection properties may be required for this connection method including:

- **Scope**: Must be specified if InitiateOAuth is set to GETANDREFRESH as the Scope is submitted to Microsoft during retrieval of credentials. This will vary depending on the service, but is generally a combination of the resource (hostname in the URL) and permission name. For example: https://host/user_impersonation.
- **AzureADTenant**: The specific Azure Tenant to authenticate against during Microsoft login. If none is specified, your user account's default tenant via the **common** login endpoint will be used. This may not be correct depending on the specific resource you are connecting to. For example, if the resource is stored on a separate tenant for cases where you have access to multiple tenants.

Otherwise, the steps are identical to the [Using OAuth Authentication](#) guide.

SharePoint Online

SharePoint Online connections may be established by retrieving a SharePoint Online cookie. Specify the following connection properties to authenticate:

- **AuthScheme**: Set this to **SharePointOnline**.
- **User**: Set this to your SharePoint Online user account.
- **Password**: Set this to your SharePoint Online password.

Kerberos

Please see [Using Kerberos](#) for details on how to authenticate with Kerberos.

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.

Logging

The adapter uses TDV Server's logging (log4j) to generate log files. The settings within the TDV Server's logging (log4j) configuration file are used by the adapter to determine the type of messages to log. The following categories can be specified:

- Error: Only error messages are logged.
- Info: Both Error and Info messages are logged.
- Debug: Error, Info, and Debug messages are 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, that is:

```
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 never logs 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 defaults to the minimum value for that particular category. For example, if Verbosity is set to a value less than 3 and the Debug category is specified, the Verbosity defaults to 3.

The following list is an explanation 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, uncomment the following line in the TDV Server's log4j.properties file (default location of this file is: C:\Program Files\TIBCO\TDV Server <version>\conf\server):

```
log4j.logger.com.cdata=DEBUG
```

The TDV Server must 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 <version>\bin. Note that reauthenticating to the TDV Studio is required after restarting the server.

Here is an example of the calls:

```
.\composite.bat monitor restart
```

All logs for the adapter are written to the "cs_server_dsrc.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 set this option, navigate to **Administrator > Configuration**. Select **Server > Configuration > Debugging** and set the Debug Output Enabled option to **True**.

Using Kerberos

This section shows how to use the adapter to authenticate using Kerberos.

Kerberos

To authenticate to OData using Kerberos, set the following properties:

- AuthScheme: Set this to **NEGOTIATE**.
- KerberosKDC: Set this to the host name or IP Address of your Kerberos KDC machine.
- KerberosRealm: Set this to **the realm of the OData Kerberos principal**. This will be the value after the '@' symbol (for instance, EXAMPLE.COM) of the **principal value** (for instance, ServiceName/MyHost@EXAMPLE.COM).
- KerberosSPN: Set this to the service and host of the OData Kerberos Principal. This is the value prior to the '@' symbol (for instance, ServiceName/MyHost) of the principal value (for instance, ServiceName/MyHost@EXAMPLE.COM).

Retrieve the Kerberos Ticket

You can use one of the following options to retrieve the required Kerberos ticket.

MIT Kerberos Credential Cache File

This option enables you to use the MIT Kerberos Ticket Manager or kinit command to get tickets. Note that you do not need to set the User or Password connection properties with this option.

1. Ensure that you have an environment variable created called **KRB5CCNAME**.
2. Set the **KRB5CCNAME** environment variable to a path pointing to your credential cache file (for instance, C:\krb_cache\krb5cc_0 or /tmp/krb5cc_0). This file is created when generating your ticket with MIT Kerberos Ticket Manager.
3. To obtain a ticket, open the MIT Kerberos Ticket Manager application, click **Get Ticket**, enter your principal name and password, then click **OK**. If successful, ticket information appears in Kerberos Ticket Manager and is stored in the credential cache file.
4. Now that you have created the credential cache file, the adapter uses the cache file to obtain the Kerberos ticket to connect to OData.

As an alternative to setting the **KRB5CCNAME** environment variable, you can directly set the file path using the KerberosTicketCache property. When set, the adapter uses the specified cache file to obtain the Kerberos ticket to connect to OData.

Keytab File

If the **KRB5CCNAME environment variable has not been set**, you can retrieve a Kerberos ticket using a Keytab File. To do so, set the User property to the desired username and set the KerberosKeytabFile property to a file path pointing to the keytab file associated with the user.

User and Password

If both the **KRB5CCNAME** environment variable and the KerberosKeytabFile property have not been set, you can retrieve a ticket using a user and password combination. To do this, set the User and Password properties to the user/password combination that you use to authenticate with OData.

Cross-Realm

More complex Kerberos environments may require cross-realm authentication where multiple realms and KDC servers are used (e.g., where one realm/KDC is used for user authentication and another realm/KDC is used for obtaining the service ticket).

In such an environment, set the KerberosRealm and KerberosKDC properties to the values required for user authentication. Also set the KerberosServiceRealm and KerberosServiceKDC properties to the values required to obtain the service ticket.

Using OAuth Authentication

OAuth requires the authenticating user to interact with OData using the browser. Most OData services will not require OAuth. This section is only for OData endpoints that require OAuth. The adapter facilitates this in various ways as described below.

Required OAuth Connection Properties

There are several required connection properties to establish an OAuth connection. These must be obtained from the API documentation of the OData service you are connecting to.

- OAuthVersion: The version of OAuth being used. Enter 1.0 or 2.0.
- OAuthAccessTokenURL: The URL to retrieve the OAuth access token from.
- OAuthAuthorizationURL: The URL that needs to be opened in the browser for the user

to grant permissions.

- OAuthRefreshTokenURL: The URL to use when refreshing an expired access token.
- OAuthRequestTokenURL: The URL the service provides to retrieve request tokens from. Required in OAuth 1.0.
- OAuthGrantType: The grant type for the OAuth flow. Can be either CLIENT, CODE or PASSWORD.
- OAuthParams: A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.

Creating a Custom OAuth App

In addition to the previously mentioned connection properties, you will need to register an app to obtain the OAuthClientId and OAuthClientSecret. See [Creating a Custom OAuth App](#) for information on creating a custom app.

Creating a Custom OAuth App

Creating a custom application in most services requires registering as a developer and creating an app in the UI of the service. This is not necessarily true for all services. In some you must contact the service provider to create the app for you. However it is done, you must obtain the values for OAuthClientId, OAuthClientSecret, and CallbackURL.

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.

Set Additional Azure AD OAuth Properties

In addition to the OAuth URLs and the following properties, set AzureResource and AzureTenant when authenticating to Azure AD OAuth endpoints.

Authenticate to OData

After setting the required URLs and the following connection properties you are ready to connect:

- OAuthVersion: Set this to 1.0 or 2.0.
- OAuthGrantType: By default, the adapter negotiates the browser-login flow. This is the "CODE" grant type. However, OAuth 2.0 also supports an exchange of login credentials for the access token; to use this grant type, set this property to "PASSWORD".
- InitiateOAuth: Set this to GETANDREFRESH. You can use InitiateOAuth to avoid repeating the OAuth exchange and manually setting the access token in the connection string.
- OAuthClientId: Set this to the client Id in your app settings. This is also called the consumer key.
- OAuthClientSecret: Set this to the client secret in your app settings. This is also called the consumer secret.
- OAuthParams: Set this to a comma-separated list of any additional parameters required by your data source.
- CallbackURL: Set this to the localhost callback url you would like to use for a response from the OAuthAuthorizationURL. We recommend using <http://localhost:33333> if possible.

Custom Credentials

Desktop Authentication with your OAuth App

Follow the steps below to authenticate with the credentials for a custom OAuth app. See [Creating a Custom OAuth App](#).

Get an OAuth Access Token

After setting the following, you are ready to connect:

- OAuthClientId: Set this to the Client Id in your app settings.
- OAuthClientSecret: Set this to the Client Secret in your app settings.
- CallbackURL: Set this to the Redirect URL in your app settings.
- 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. Obtains a new access token when the old one expires.
3. Saves OAuth values in OAuthSettingsLocation to be persisted across connections.

Headless Machines

Using OAuth on a Headless Machine

To create OData data sources on headless servers or other machines on which the adapter cannot open a browser, 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 Credentials for a Custom OAuth App

Create a Custom OAuth App

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 App Id in your app settings.
- OAuthClientSecret: Set this to the App 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. Save the value of the returned AuthToken and AuthKey if OAuthVersion is set to 1.0. They will be used in the next step.
3. 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.
4. 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:

- OAuthClientId: Set this to the consumer key in your app settings.
- OAuthClientSecret: Set this to the consumer secret in your app settings.
- OAuthVerifier: Set this to the verifier code.
- AuthToken: If OAuthVersion is set to 1.0, set this to the AuthToken.
- AuthKey: If OAuthVersion is set to 1.0, set this to the AuthKey.
- OAuthSettingsLocation: Set this to persist the encrypted OAuth authentication values to the specified file.
- InitiateOAuth: Set this to REFRESH.

Connect to Data

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 in your app settings.
- OAuthClientSecret: Set this to the Client Secret in your app settings.
- CallbackURL: Set this to the Callback URL in your app settings.

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.

Fine-Tuning Data Access

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.

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.

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.
- ServerTimeZone: By default, the adapter assumes the server's Edm.DateTime values are GMT and converts them to and from the installed machine's local timezone accordingly.

If the server's Edm.DateTime values are known to apply to a different timezone, then set this property to that timezone (i.e. EST).

Changelog

General Changes

Date	Build Number	Change Type	Description
12/14/2022	8383	General	Changed <ul style="list-style-type: none"> • Added the Default column to the sys_procedureparameters table.
09/30/2022	8308	General	Changed <ul style="list-style-type: none"> • Added the IsPath column to the sys_procedureparameters table.
08/17/2022	8264	General	Changed <ul style="list-style-type: none"> • We now support handling the keyword "COLLATE" as standard function name as well.
03/09/ [8103] 2022	8103	OData	Changed <ul style="list-style-type: none"> • Changed the default method for submitting client_credentials grants with OAuth. Before client_credentials

			<p>would submit OAuthClientId and OAuthClientSecret using the HTTP Authorization header. Now the OAuthPasswordGrantType controls this setting and it defaults to OAuthPasswordGrantType=Post. The previous behavior is still available by setting OAuthPasswordGrantType=Basic.</p>
02/16/2022	8083	OData	<p>Added</p> <ul style="list-style-type: none"> Added support to control whether the LIKE operator is sent server-side or processed client-side through the hidden connection property, SupportsLike. This is set by default to True, and when changed to False, the LIKE operator will be executed client-side.
09/02/2021	7915	General	<p>Added</p> <ul style="list-style-type: none"> Added support for the STRING_SPLIT table-valued function in the CROSS APPLY clause.
08/07/2021	7889	General	<p>Changed</p> <ul style="list-style-type: none"> Added the KeySeq column to the sys_foreignkeys table.
08/06/2021	7888	General	<p>Changed</p> <ul style="list-style-type: none"> Added the new sys_primarykeys system table.
07/23/2021	7874	General	<p>Changed</p> <ul style="list-style-type: none"> Updated the Literal Function Names for relative date/datetime functions. Previously relative date/datetime

			<p>functions resolved to a different value when used in the projection vs the predicate. I.e: SELECT LAST_MONTH() AS lm, Col FROM Table WHERE Col > LAST_MONTH(). Formerly the two LAST_MONTH() methods would resolve to different datetimes. Now they will match.</p> <ul style="list-style-type: none"> As a replacement for the previous behavior, the relative date/datetime functions in the criteria may have an 'L' appended to them. I.e: WHERE col > L_LAST_MONTH(). This will continue to resolve to the same values that previously were calculated in the criteria. Note that the "L_" prefix will only work in the predicate - it not available for the projection.
07/08/2021	7859	General	<p>Added</p> <ul style="list-style-type: none"> Added the TCP Logging Module for the logging information happening on the TCP wire protocol. The transport bytes that are incoming and ongoing will be logged at verbosity=5.
09/21/2021	7830	OData	<p>Changed</p> <ul style="list-style-type: none"> Updated the stored procedures ListAssociations, CreateAssociation and RemoveAssociation to include UrlId (to bypass fromId) and ToUrlId (to bypass toId) in cases where the primary key consists of multiple attributes.
04/23/2021	7785	General	<p>Added</p>

			<ul style="list-style-type: none"> Added support for handling client side formulas during insert / update. For example: UPDATE Table SET Col1 = Concat(Col1, " - ", Col2) WHERE Col2 LIKE 'A%'
04/23/2021	7783	General	<p>Changed</p> <ul style="list-style-type: none"> Updated how display sizes are determined for varchar primary key and foreign key columns so they will match the reported length of the column.
04/16/2021	7776	General	<p>Added</p> <ul style="list-style-type: none"> Non-conditional updates between two columns is now available to all drivers. For example: UPDATE Table SET Col1=Col2 <p>Changed</p> <ul style="list-style-type: none"> Reduced the length to 255 for varchar primary key and foreign key columns. Updated implicit and metadata caching to improve performance and support for multiple connections. Old metadata caches are not compatible - you would need to generate new metadata caches if you are currently using CacheMetadata. Updated index naming convention to avoid duplicates Updated and standardized Getting Started connection help. Added the Advanced Features section to the help of all drivers.

			<ul style="list-style-type: none"> • Categorized connection property listings in the help for all editions.
04/15 /2021	7775	General	Changed <ul style="list-style-type: none"> • Kerberos authentication is updated to use TCP by default, but will fall back to UDP if a TCP connection cannot be established
02/22/2021	7723	OData	Added <ul style="list-style-type: none"> • Added support for non-atomic batch operations. This behavior can be achieved by specifying the EnableAtomicBatchOperations connection property.
01/14/2021	7684	OData	Added <ul style="list-style-type: none"> • Added support for a direct connection to OData feeds. The user must specify the FeedURL connection property for this to work.
01/12/2021	7682	OData	Added <ul style="list-style-type: none"> • Added support for allowing /\$metadata endpoints to be used directly for the url instead of the service document. Some OData services may only support /\$metadata instead of direct service document listing.
12/06/2020	7645	OData	Added <ul style="list-style-type: none"> • Added support for listing stored procedures which return collection of entities as views.
11/10/2020	7619	OData	Added

- Added support for parsing dynamic stored procedures with primitive return types.
-

Advanced Features

This section details a selection of advanced features of the OData adapter.

User Defined Views

The adapter allows you to define virtual tables, called *user defined views*, whose contents are decided by a pre-configured query. These views are useful when you cannot directly control queries being issued to the drivers. See [User Defined Views](#) for an overview of creating and configuring custom views.

SSL Configuration

Use [SSL Configuration](#) to adjust how adapter handles TLS/SSL certificate negotiations. You can choose from various certificate formats; see the [SSLServerCert](#) property under "Connection String Options" for more information.

Firewall and Proxy

Configure the adapter for compliance with [Firewall and Proxy](#), including Windows proxies and HTTP proxies. You can also set up tunnel connections.

Query Processing

The adapter offloads as much of the SELECT statement processing as possible to OData and then processes the rest of the query in memory (client-side).

See [Query Processing](#) for more information.

Logging

See [Logging](#) for an overview of configuration settings that can be used to refine CData logging. For basic logging, you only need to set two connection properties, but there are numerous features that support more refined logging, where you can select subsets of information to be logged using the [LogModules](#) connection property.

User Defined Views

The OData Adapter allows you to define a virtual table whose contents are decided by a pre-configured query. These are called *User Defined Views*, which are useful in situations where you cannot directly control the query being issued to the driver, e.g. when using the driver from a tool. The User Defined Views can be used to define predicates that are always applied. If you specify additional predicates in the query to the view, they are combined with the query already defined as part of the view.

There are two ways to create user defined views:

- Create a JSON-formatted configuration file defining the views you want.
- DDL statements.

Defining Views Using a Configuration File

User Defined Views are defined in a JSON-formatted configuration file called *UserDefinedViews.json*. The adapter automatically detects the views specified in this file.

You can also have multiple view definitions and control them using the [UserDefinedViews](#) connection property. When you use this property, only the specified views are seen by the adapter.

This User Defined View configuration file is formatted as follows:

- Each root element defines the name of a view.
- Each root element contains a child element, called **query**, which contains the custom SQL query for the view.

For example:

```
{
  "MyView": {
    "query": "SELECT * FROM Lead WHERE MyColumn = 'value'"
  }
}
```

```

    },
    "MyView2": {
      "query": "SELECT * FROM MyTable WHERE Id IN (1,2,3)"
    }
  }
}

```

Use the `UserDefinedViews` connection property to specify the location of your JSON configuration file. For example:

```

"UserDefinedViews",
"C:\\Users\\yourusername\\Desktop\\tmp\\UserDefinedViews.json"

```

Defining Views Using DDL Statements

The adapter is also capable of creating and altering the schema via DDL Statements such as CREATE LOCAL VIEW, ALTER LOCAL VIEW, and DROP LOCAL VIEW.

Create a View

To create a new view using DDL statements, provide the view name and query as follows:

```
CREATE LOCAL VIEW [MyViewName] AS SELECT * FROM Customers LIMIT 20;
```

If no JSON file exists, the above code creates one. The view is then created in the JSON configuration file and is now discoverable. The JSON file location is specified by the `UserDefinedViews` connection property.

Alter a View

To alter an existing view, provide the name of an existing view alongside the new query you would like to use instead:

```
ALTER LOCAL VIEW [MyViewName] AS SELECT * FROM Customers WHERE
TimeModified > '3/1/2020';
```

The view is then updated in the JSON configuration file.

Drop a View

To drop an existing view, provide the name of an existing schema alongside the new query you would like to use instead.

```
DROP LOCAL VIEW [MyViewName]
```

This removes the view from the JSON configuration file. It can no longer be queried.

Schema for User Defined Views

User Defined Views are exposed in the **UserViews** schema by default. This is done to avoid the view's name clashing with an actual entity in the data model. You can change the name of the schema used for UserViews by setting the UserViewsSchemaName property.

Working with User Defined Views

For example, a SQL statement with a User Defined View called *UserViews.RCustomers* only lists customers in Raleigh:

```
SELECT * FROM Customers WHERE City = 'Raleigh';
```

An example of a query to the driver:

```
SELECT * FROM UserViews.RCustomers WHERE Status = 'Active';
```

Resulting in the effective query to the source:

```
SELECT * FROM Customers WHERE City = 'Raleigh' AND Status = 'Active';
```

That is a very simple example of a query to a User Defined View that is effectively a combination of the view query and the view definition. It is possible to compose these queries in much more complex patterns. All SQL operations are allowed in both queries and are combined when appropriate.

Inserting Parent and Child Records

Use Case

When inserting records, often there is a need to fill in details about child records that have a dependency on a parent.

For instance, when dealing with a CRM system, Invoices often may not be entered without at least one line item.

Invoices may have many line items, with each line item made up of several fields. This presents a unique challenge when offering the data as relational tables.

When reading the data, it is easy enough to model an Invoice and an InvoiceLineItem table with a foreign key connecting the two.

But during inserts, the CRM system will require both the Invoice and the InvoiceLineItems to be created in a single submission.

To solve this sort of problem, our tools offer child collection columns on the parent. These columns may be used to submit insert statements that include details of both the parent and the child records.

From our example, the Invoice table may have a column called InvoiceLineItems. During the insert, we can pass in the details of the records that would need to be inserted to the InvoiceLineItems table into the InvoiceLineItems column of the Invoice record. This can be done using the following methods:

Methods for Inserting Parent/Child Records

The adapter facilitates two methods for inserting parent/child records: temporary table insertion and XML/JSON aggregate insertion.

Temporary (#TEMP) tables

The simplest way to enter data would be to use a #TEMP table, or temporary table, which the adapter will store in memory.

Reference the #TEMP table with the following syntax:

TableName#TEMP

#TEMP tables are stored in memory for the duration of a connection.

Therefore, in order to use them, you cannot close the connection between submitting inserts to them, and they cannot be used in environments where a different connection may be used for each query.

Within that single connection, the table remains in memory until the bulk insert is successful, at which point the temporary table will be wiped from memory.

For example:

```
INSERT INTO InvoiceLineItems#TEMP (ReferenceNumber, Item, Quantity,
Amount) VALUES ('INV001', 'Basketball', 10, 9.99)
INSERT INTO InvoiceLineItems#TEMP (ReferenceNumber, Item, Quantity,
Amount) VALUES ('INV001', 'Football', 5, 12.99)
```

Once the InvoiceLineItems table is populated, the #TEMP table may be referenced during an insert into the Invoice table:

```
INSERT INTO Invoices (ReferenceNumber, Customer, InvoiceLines) VALUES
('INV001', 'John Doe', 'InvoiceLineItems#TEMP')
```

Under the hood, the adapter will read in values from the #TEMP table.

Notice that the ReferenceNumber was used to identify what Invoice the lines are tied to. This is because the #TEMP table may be populated and used with a bulk insert, where you will have different lines for different Invoices.

This allows the #TEMP tables to be used with a bulk insert. For example:

```
INSERT INTO Invoices#TEMP (ReferenceNumber, Customer, InvoiceLines)
VALUES ('INV001', 'John Doe', 'InvoiceLineItems#TEMP')
INSERT INTO Invoices#TEMP (ReferenceNumber, Customer, InvoiceLines)
VALUES ('INV002', 'Jane Doe', 'InvoiceLineItems#TEMP')
INSERT INTO Invoices SELECT ReferenceNumber, Customer, InvoiceLines FROM
Invoices#TEMP
```

In this case, we are inserting two different Invoices. The ReferenceNumber is how we determine which Lines go with which Invoice.

Note: The tables and columns presented here are an example of how the adapter works in general. The specific table and column names may be different in the adapter.

XML/JSON Aggregates

As an alternative to #TEMP tables, direct XML/JSON may be used. Since #TEMP tables are not used to construct them, it does not matter if you use the same connection or close the connection after insert.

For example:

```
[
  {
    "Item", "Basketball",
```

```

    "Quantity": 10
    "Amount": 9.99
  },
  {
    "Item", "Football",
    "Quantity": 5
    "Amount": 12.99
  }
]

```

OR

```

<Row>
  <Item>Basketball</Item>
  <Quantity>10</Quantity>
  <Amount>9.99</Amount>
</Row>
<Row>
  <Item>Football</Item>
  <Quantity>5</Quantity>
  <Amount>12.99</Amount>
</Row>

```

Note that the ReferenceNumber is not present in these examples.

That is because the XML/JSON by its nature is not being passed by reference, but passed in full per insert against the parent record.

There is no need to provide something to tie the child back to the parent since the complete XML/JSON must be constructed and submitted for each row.

Then, insert the values:

```

INSERT INTO Invoices (ReferenceNumber, Customer, InvoiceLines) VALUES
('INV001', 'John Doe', '{...}')

```

OR

```

INSERT INTO Invoices (ReferenceNumber, Customer, InvoiceLines) VALUES
('INV001', 'John Doe', '<Row>...</Row>')

```

Executing Deep Inserts with SQL

The adapter supports OData deep inserts, in which you simultaneously create a base entity and link it to related entities, by specifying navigation properties. To specify [Navigation Properties](#) for an entity, you may either submit JSON / XML data, or you may create a temporary table for the navigation property and then reference the temporary table in the insert to the base table. Submit the XML / JSON or reference the temporary table in the appropriate navigation property column on the base table. Each navigation property column is prefixed with the word "Linked".

Example: Deep Inserts using XML / JSON

To submit XML or JSON data, simply supply the values for the table the navigation property is referencing in XML or JSON format. If you are familiar with the OData standard, you should not be submitting values in the standard. The XML / JSON used here is simply a means of supplying multiple values to the OData Adapter.

For example, consider the Orders table in Northwind odata.org test service. To create a new Order, you specify the Products ordered, Customer, Employee, and Shipper. To do so, you need to specify the Customer, Order_Details, Shipper, and Employee navigation properties.

- **Customer:** The following XML represents a new Customer:

```
<Row>
  <CustomerID>VINET</CustomerID>
  <CompanyName>Vins et alcools Chevalier</CompanyName>
  <ContactName>Paul Henriot</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>59 rue de l'Abbaye</Address>
  <City>Reims</City>
  <PostalCode>51100</PostalCode>
  <Country>France</Country>
  <Phone>26.47.15.10</Phone>
  <Fax>26.47.15.11</Fax>
</Row>
```

- **Order_Details:** The following JSON add two Products to the Order:

```
[
  {
```

```

        "ProductID": 72,
        "UnitPrice": 34.80,
        "Quantity": 5,
        "Discount": 0
      },
      {
        "ProductID": 42,
        "ProductID": 9.80,
        "ProductID": 10,
        "ProductID": 0
      }
    ]

```

- **Employee:** The following XML specifies the existing Employee:

```

<Row>
  <EmployeeID>5</EmployeeID>
</Row>

```

- **Shipper:** The following JSON specifies the existing Shipper:

```

[
  {
    "ShipperID": 3
  }
]

```

In order to execute the insert, simply reference or include as string literals the complete XML / JSON. For example:

```

INSERT INTO Orders (CustomerID, EmployeeID, ShipVia, ShipName,
ShipAddress, ShipCity, ShipPostalCode, ShipCountry, OrderDate,
LinkedOrder_Details, LinkedCustomer, LinkedEmployee, LinkedShipper)
VALUES ('VINET', 5, 3, 'Paul Henriot', '59 rue de l'Abbaye', 'Reims',
'51100', 'France', '07/04/1996', '{ ... }', '<Row>...</Row>', ?, ?)

```

Example: Deep Inserts using Temporary Tables

If using temporary tables, they must be defined and inserted within the same connection. Closing the connection will clear out any temporary tables in memory. Keeping with the Northwind example, you need to specify the following navigation properties.

Creating Temporary Tables

Insert the related entities into temporary tables that correspond to each navigation property. You can specify an existing entity's primary key or you can insert a new entity.

- **Customer:** The following statement creates a new Customer:

```
INSERT INTO Customers#TEMP (CustomerID, CompanyName, ContactName,
ContactTitle, Address, City, PostalCode, Country, Phone, Fax)
VALUES ('VINET', 'Vins et alcools Chevalier', 'Paul Henriot',
'Accounting Manager', '59 rue de l'Abbaye', 'Reims', '51100',
'France', '26.47.15.10', '26.47.15.11')
```

- **Order Details:** The following statements add two Products to the Order:

```
INSERT INTO Order_Details#TEMP (ProductID, UnitPrice, Quantity,
Discount) VALUES (72, 34.80, 5, 0)
INSERT INTO Order_Details#TEMP (ProductID, UnitPrice, Quantity,
Discount) VALUES (42, 9.80, 10, 0)
```

- **Employee:** The following statement specifies the existing Employee:

```
INSERT INTO Employees#TEMP (EmployeeID)
VALUES (5)
```

- **Shipper:** The following statement specifies the existing Shipper:

```
INSERT INTO Shippers#TEMP (ShipperID) VALUES (3)
```

The OData Adapter will assume that the Shipper and Employee already exist and will only link to the existing references since only the primary keys were specified for either. When more than just the primary key is defined, such as the examples for Customer and Order_Details, the OData Adapter will attempt to create new entries - triggering the deep insert.

Inserting the Entity

In the INSERT statement for the base entity, reference the temporary tables in the LinkedOrder_Details, LinkedCustomer, LinkedEmployee, and LinkedShipper columns:

```
INSERT INTO Orders (CustomerID, EmployeeID, ShipVia, ShipName,
ShipAddress, ShipCity, ShipPostalCode, ShipCountry, OrderDate,
LinkedOrder_Details, LinkedCustomer, LinkedEmployee, LinkedShipper)
VALUES ('VINET', 5, 3, 'Paul Henriot', '59 rue de l'Abbaye', 'Reims',
'51100', 'France', '07/04/1996', 'Order_Details#TEMP', 'Customers#TEMP',
'Employees#TEMP', 'Shippers#TEMP')
```

Code Example

Below is the complete code to create the new Order:

```
Connection conn = DriverManager.getConnection
("jdbc:odata:URL=http://services.odata.org/Northwind/Northwind.svc;");
Statement stat = conn.createStatement();
stat.executeUpdate("INSERT INTO Customers#TEMP (CustomerID, CompanyName,
ContactName, ContactTitle, Address, City, PostalCode, Country, Phone,
Fax) VALUES ('VINET', 'Vins et alcools Chevalier', 'Paul Henriot',
'Accounting Manager', '59 rue de l'Abbaye', 'Reims', '51100', 'France',
'26.47.15.10', '26.47.15.11')");
stat.executeUpdate("INSERT INTO Order_Details#TEMP (ProductID,
UnitPrice, Quantity, Discount) VALUES (72, 34.80, 5, 0)");
stat.executeUpdate("INSERT INTO Order_Details#TEMP (ProductID,
UnitPrice, Quantity, Discount) VALUES (42, 9.80, 10, 0)");
stat.executeUpdate("INSERT INTO Employees#TEMP (EmployeeID) VALUES
(5)");
stat.executeUpdate("INSERT INTO Shippers#TEMP (ShipperID) VALUES (3)");
stat.executeUpdate("INSERT INTO Orders (CustomerID, EmployeeID, ShipVia,
ShipName, ShipAddress, ShipCity, ShipPostalCode, ShipCountry, OrderDate,
LinkedOrder_Details, LinkedCustomer, LinkedEmployee, LinkedShipper)
VALUES ('VINET', 5, 3, 'Paul Henriot', '59 rue de l'Abbaye', 'Reims',
'51100', 'France', '07/04/1996', 'Order_Details#TEMP', 'Customers#TEMP',
'Employees#TEMP', 'Shippers#TEMP')");
stat.close();
```

Example: Bulk Inserts using Temporary Tables

If using temporary tables, they must be defined and inserted within the same connection. Closing the connection will clear out any temporary tables in memory. Use [IncludeReferenceColumn](#) connection property to add a input only Reference column to properly associate children during a deep insert with the same parent. Keeping with the Northwind example, you need to specify the following navigation properties.

Creating Temporary Tables

Insert the related entities into temporary tables that correspond to each navigation property. You can specify an existing entity's primary key or you can insert a new entity.

- **Categories:** The following statements adds to Categories child table:

```
INSERT INTO Categories#TEMP (ParentReference, ID, Name) VALUES
(100, 4, 'DVD')
INSERT INTO Categories#TEMP (ParentReference, ID, Name) VALUES
(100, 5, 'BluRay')
INSERT INTO Categories#TEMP (ParentReference, ID, Name) VALUES
(200, 6, 'Radio')
```

- **Products:** The following statements adds to Product parent table:

```
INSERT INTO Products#TEMP (ID, ParentReference, price,
CategoriesAggregate) VALUES (20, 100, 45, 'Categories#TEMP')
INSERT INTO Products#TEMP (ID, ParentReference, price,
CategoriesAggregate) VALUES (21, 200, 25, 'Categories#TEMP')
```

Inserting the Entity

In the INSERT statement for the base entity, reference the temporary table in the LinkedCategories column:

```
INSERT INTO Products (ID, ParentReference, price, LinkedCategories)
SELECT ID, ParentReference, price, CategoriesAggregate FROM
Products#TEMP
```

Code Example

Below is the complete code to insert into Products:

```
Connection conn = DriverManager.getConnection
("jdbc:odata:URL=http://services.odata.org/Northwind/Northwind.svc;Inclu
deReferenceColumn=true;");
Statement stat = conn.createStatement();
stat.executeUpdate("INSERT INTO Categories#TEMP (ParentReference, ID,
Name) VALUES (100, 4, 'DVD')");
stat.executeUpdate("INSERT INTO Categories#TEMP (ParentReference, ID,
```

```

Name) VALUES (100, 5, 'BluRay'))";
stat.executeUpdate("INSERT INTO Categories#TEMP (ParentReference, ID,
Name) VALUES (200, 6, 'Radio')");
stat.executeUpdate("INSERT INTO Products#TEMP (ID, ParentReference,
price, CategoriesAggregate) VALUES (20, 100, 45, 'Categories#TEMP')");
stat.executeUpdate("INSERT INTO Products#TEMP (ID, ParentReference,
price, CategoriesAggregate) VALUES (21, 200, 25, 'Categories#TEMP')");
stat.executeUpdate("INSERT INTO Products (ID, ParentReference, price,
LinkedCategories) SELECT ID, ParentReference, price, CategoriesAggregate
FROM Products#TEMP");
stat.close();

```

SSL Configuration

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.

Client SSL Certificates

The OData adapter also supports setting client certificates. Set the following to connect using a client certificate.

- [SSLClientCert](#): The name of the certificate store for the client certificate.
- [SSLClientCertType](#): The type of key store containing the TLS/SSL client certificate.
- [SSLClientCertPassword](#): The password for the TLS/SSL client certificate.
- [SSLClientCertSubject](#): The subject of the TLS/SSL client certificate.

Firewall and Proxy

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.

In addition, to authenticate to an HTTP proxy, set ProxyAuthScheme, ProxyUser, and ProxyPassword, in addition to ProxyServer and ProxyPort.

Other Proxies

Set the following properties:

- To use a proxy-based firewall, 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.

Query Processing

Query Processing

CData has a client-side SQL engine built into the adapter library. This enables support for the full capabilities that SQL-92 offers, including filters, aggregations, functions, etc.

For sources that do not support SQL-92, the adapter offloads as much of SQL statement processing as possible to OData and then processes the rest of the query in memory (client-side). This results in optimal performance.

For data sources with limited query capabilities, the adapter handles transformations of the SQL query to make it simpler for the adapter. The goal is to make smart decisions based on the query capabilities of the data source to push down as much of the computation as possible. The OData Query Evaluation component examines SQL queries and returns information indicating what parts of the query the adapter is not capable of executing natively.

The OData Query Slicer component is used in more specific cases to separate a single query into multiple independent queries. The client-side Query Engine makes decisions

about simplifying queries, breaking queries into multiple queries, and pushing down or computing aggregations on the client-side while minimizing the size of the result set.

There's a significant trade-off in evaluating queries, even partially, client-side. There are always queries that are impossible to execute efficiently in this model, and some can be particularly expensive to compute in this manner. CData always pushes down as much of the query as is feasible for the data source to generate the most efficient query possible and provide the most flexible query capabilities.

More Information

For a full discussion of how CData handles query processing, see [CData Architecture: Query Execution](#).

Logging

Capturing adapter logging can be very helpful when diagnosing error messages or other unexpected behavior.

Basic Logging

You will simply need to set two connection properties to begin capturing adapter logging.

- Logfile: A filepath which designates the name and location of the log file.
- Verbosity: This is a numerical value (1-5) that determines the amount of detail in the log. See the page in the Connection Properties section for an explanation of the five levels.
- MaxLogFileSize: When the limit is hit, a new log is created in the same folder with the date and time appended to the end. The default limit is 100 MB. Values lower than 100 kB will use 100 kB as the value instead.
- MaxLogFileCount: A string specifying the maximum file count of log files. When the limit is hit, a new log is created in the same folder with the date and time appended to the end and the oldest log file will be deleted. Minimum supported value is 2. A value of 0 or a negative value indicates no limit on the count.

Once this property is set, the adapter will populate the log file as it carries out various tasks, such as when authentication is performed or queries are executed. If the specified file doesn't already exist, it will be created.

Log Verbosity

The verbosity level determines the amount of detail that the adapter reports to the [Logfile](#). [Verbosity](#) levels from 1 to 5 are supported. These are described in the following list:

1	Setting Verbosity to 1 will log the query, the number of rows returned by it, the start of execution and the time taken, and any errors.
2	Setting Verbosity to 2 will log everything included in Verbosity 1 and additional information about the request.
3	Setting Verbosity to 3 will additionally log HTTP headers, as well as the body of the request and the response.
4	Setting Verbosity to 4 will additionally log transport-level communication with the data source. This includes SSL negotiation.
5	Setting Verbosity to 5 will additionally log communication with the data source and additional details that may be helpful in troubleshooting problems. This includes interface commands.

The [Verbosity](#) should not be set to greater than 1 for normal operation. Substantial amounts of data can be logged at higher verbosity levels, which can delay execution times.

To refine the logged content further by showing/hiding specific categories of information, see [LogModules](#).

Sensitive Data

Verbosity levels 3 and higher may capture information that you do not want shared outside of your organization. The following lists information of concern for each level:

- Verbosity 3: The full body of the request and the response, which includes all the data returned by the adapter
- Verbosity 4: SSL certificates
- Verbosity 5: Any extra transfer data not included at Verbosity 3, such as non human-readable binary transfer data

Best Practices for Data Security

Although we mask sensitive values, such as passwords, in the connection string and any request in the log, it is always best practice to review the logs for any sensitive information before sharing outside your organization.

Java Logging

When Java logging is enabled in Logfile, the Verbosity will instead map to the following logging levels.

- 0: Level.WARNING
- 1: Level.INFO
- 2: Level.CONFIG
- 3: Level.FINE
- 4: Level.FINER
- 5: Level.FINEST

Advanced Logging

You may want to refine the exact information that is recorded to the log file. This can be accomplished using the LogModules property.

This property allows you to filter the logging using a semicolon-separated list of logging modules.

All modules are four characters long. **Please note that modules containing three letters have a required trailing blank space.** The available modules are:

- **EXEC:** Query Execution. Includes execution messages for original SQL queries, parsed SQL queries, and normalized SQL queries. Query and page success/failure messages appear here as well.
- **INFO:** General Information. Includes the connection string, driver version (build number), and initial connection messages.
- **HTTP:** HTTP Protocol messages. Includes HTTP requests/responses (including POST messages), as well as Kerberos related messages.
- **SSL :** SSL certificate messages.
- **OAUT:** OAuth related failure/success messages.

- **SQL** : Includes SQL transactions, SQL bulk transfer messages, and SQL result set messages.
- **META**: Metadata cache and schema messages.
- **TCP** : Incoming and Ongoing raw bytes on TCP transport layer messages.

An example value for this property would be.

```
LogModules=INFO;EXEC;SSL ;SQL ;META;
```

Note that these modules refine the information as it is pulled after taking the Verbosity into account.

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.

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-Z, a-z, 0-9, _:@].
- 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 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 ] [ NULLS FIRST | NULLS LAST ]
]
[
  LIMIT <expression>
  [
    { OFFSET | , }
    <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> { = | != | > | < | >= | <= | IN | IS NULL | IS NOT
NULL | LIKE | AND | OR } [ <expression> ]
    } [ { AND | OR } ... ]

```

Examples

1. Return all columns:

```
SELECT * FROM Lead
```

2. Rename a column:

```
SELECT "FullName" AS MY_FullName FROM Lead
```

3. Cast a column's data as a different data type:

```
SELECT CAST(AnnualRevenue AS VARCHAR) AS Str_AnnualRevenue FROM Lead
```

4. Search data:

```
SELECT * FROM Lead WHERE FirstName <> 'Bartholomew'
```

5. The OData APIs support the following operators in the WHERE clause: =, !=, >, <, >=, <=, IN, IS NULL, IS NOT NULL, LIKE, AND, OR.

```
SELECT * FROM Lead WHERE FirstName <> 'Bartholomew';
```

6. Return the number of items matching the query criteria:

```
SELECT COUNT(*) AS MyCount FROM Lead
```

7. 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_search, 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_search, 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):

```
boolean ret = stat.execute("SELECT Id, FullName INTO 'csv://c:/Lead.txt'
FROM 'Lead' WHERE FirstName <> 'Bartholomew'");
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 'Lead' IN
'csv://filename=c:/Lead.csv;delimiter=tab' FROM 'Lead' WHERE FirstName
<> 'Bartholomew'");
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, as shown in the following example:

```
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, as shown in the following example:

```
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, as shown in the following example:

```
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 information 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, as shown in the following example:

```
<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, as shown in the following example:

```
Connection connection = DriverManager.getConnection
("jdbc:odata:User=myuseraccount;Password=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;
```

PIVOT and UNPIVOT

PIVOT and **UNPIVOT** can be used to change a table-valued expression into another table.

PIVOT

PIVOT rotates a table-value expression by turning unique values from one column into multiple columns in the output. PIVOT can run aggregations where required on any column value.

PIVOT Syntax

```
"SELECT 'AverageCost' AS Cost_Sorted_By_Production_Days, [0], [1], [2],  
[3], [4]  
FROM  
(  
  SELECT DaysToManufacture, StandardCost  
  FROM Production.Product  
) AS SourceTable  
PIVOT  
(  
  AVG(StandardCost)  
  FOR DaysToManufacture IN ([0], [1], [2], [3], [4])  
) AS PivotTable;"
```

UNPIVOT

UNPIVOT carries out nearly the opposite to PIVOT by rotating columns of a table-valued expressions into column values.

UNPIVOT Sytax

```
"SELECT VendorID, Employee, Orders
FROM
(SELECT VendorID, Emp1, Emp2, Emp3, Emp4, Emp5
FROM pvt) p
UNPIVOT
(Orders FOR Employee IN
(Emp1, Emp2, Emp3, Emp4, Emp5)
)AS unpvt;"
```

For further information on PIVOT and UNPIVOT, see [FROM clause plus JOIN, APPLY, PIVOT \(Transact-SQL\)](#)

Data Model

The OData Adapter models OData entities in relational Tables, Views, and Stored Procedures. The table definitions are dynamically obtained from the OData service you connect to. Any changes in the metadata, such as added or removed columns or changes in data type, can be loaded by reconnecting.

Tables

The adapter models the writable entity sets and singletons described in the service metadata document as bidirectional [Tables](#).

Views

Some OData entities can only be accessed through [Navigation Properties](#). By default, the adapter models navigation properties as separate views. You can disable this behavior with [NavigationPropertiesAsViews](#). See [Views](#) for more information on querying navigation properties.

Stored Procedures

[Stored Procedures](#) are function-like interfaces to the data source. They can be used to search, update, and modify information in the data source.

Tables

The adapter exposes tables for every entity set and singleton defined on the OData service document. Entities on these tables may be inserted, updated, or deleted using standard SQL insert, update, or delete statements.

Executing Deep Inserts with SQL

The adapter supports OData deep inserts, in which you simultaneously create a base entity and link it to related entities, by specifying navigation properties. To specify [Navigation Properties](#) for an entity, you may either submit JSON / XML data, or you may create a temporary table for the navigation property and then reference the temporary table in the insert to the base table. Submit the XML / JSON or reference the temporary table in the appropriate navigation property column on the base table. Each navigation property column is prefixed with the word "Linked".

Example: Deep Inserts using XML / JSON

To submit XML or JSON data, simply supply the values for the table the navigation property is referencing in XML or JSON format. If you are familiar with the OData standard, you should not be submitting values in the standard. The XML / JSON used here is simply a means of supplying multiple values to the OData Adapter.

For example, consider the Orders table in Northwind odata.org test service. To create a new Order, you specify the Products ordered, Customer, Employee, and Shipper. To do so, you need to specify the Customer, Order_Details, Shipper, and Employee navigation properties.

- **Customer:** The following XML represents a new Customer:

```
<Row>
  <CustomerID>VINET</CustomerID>
    <CompanyName>Vins et alcools Chevalier</CompanyName>
    <ContactName>Paul Henriot</ContactName>
```

```

    <ContactTitle>Accounting Manager</ContactTitle>
    <Address>59 rue de l'Abbaye</Address>
    <City>Reims</City>
    <PostalCode>51100</PostalCode>
    <Country>France</Country>
    <Phone>26.47.15.10</Phone>
    <Fax>26.47.15.11</Fax>
  </Row>

```

- **Order_Details:** The following JSON add two Products to the Order:

```

[
  {
    "ProductID": 72,
    "UnitPrice": 34.80,
    "Quantity": 5,
    "Discount": 0
  },
  {
    "ProductID": 42,
    "ProductID": 9.80,
    "ProductID": 10,
    "ProductID": 0
  }
]

```

- **Employee:** The following XML specifies the existing Employee:

```

<Row>
  <EmployeeID>5</EmployeeID>
</Row>

```

- **Shipper:** The following JSON specifies the existing Shipper:

```

[
  {
    "ShipperID": 3
  }
]

```

In order to execute the insert, simply reference or include as string literals the complete XML / JSON. For example:

```
INSERT INTO Orders (CustomerID, EmployeeID, ShipVia, ShipName,
ShipAddress, ShipCity, ShipPostalCode, ShipCountry, OrderDate,
LinkedOrder_Details, LinkedCustomer, LinkedEmployee, LinkedShipper)
VALUES ('VINET', 5, 3, 'Paul Henriot', '59 rue de l'Abbaye', 'Reims',
'51100', 'France', '07/04/1996', '{ ... }', '<Row>...</Row>', ?, ?)
```

Example: Deep Inserts using Temporary Tables

If using temporary tables, they must be defined and inserted within the same connection. Closing the connection will clear out any temporary tables in memory. Keeping with the Northwind example, you need to specify the following navigation properties.

Creating Temporary Tables

Insert the related entities into temporary tables that correspond to each navigation property. You can specify an existing entity's primary key or you can insert a new entity.

- **Customer:** The following statement creates a new Customer:

```
INSERT INTO Customers#TEMP (CustomerID, CompanyName, ContactName,
ContactTitle, Address, City, PostalCode, Country, Phone, Fax)
VALUES ('VINET', 'Vins et alcools Chevalier', 'Paul Henriot',
'Accounting Manager', '59 rue de l'Abbaye', 'Reims', '51100',
'France', '26.47.15.10', '26.47.15.11')
```

- **Order Details:** The following statements add two Products to the Order:

```
INSERT INTO Order_Details#TEMP (ProductID, UnitPrice, Quantity,
Discount) VALUES (72, 34.80, 5, 0)
INSERT INTO Order_Details#TEMP (ProductID, UnitPrice, Quantity,
Discount) VALUES (42, 9.80, 10, 0)
```

- **Employee:** The following statement specifies the existing Employee:

```
INSERT INTO Employees#TEMP (EmployeeID)
VALUES (5)
```

- **Shipper:** The following statement specifies the existing Shipper:

```
INSERT INTO Shippers#TEMP (ShipperID) VALUES (3)
```

The OData Adapter will assume that the Shipper and Employee already exist and will only link to the existing references since only the primary keys were specified for either. When more than just the primary key is defined, such as the examples for Customer and Order_Details, the OData Adapter will attempt to create new entries - triggering the deep insert.

Inserting the Entity

In the INSERT statement for the base entity, reference the temporary tables in the LinkedOrder_Details, LinkedCustomer, LinkedEmployee, and LinkedShipper columns:

```
INSERT INTO Orders (CustomerID, EmployeeID, ShipVia, ShipName,
ShipAddress, ShipCity, ShipPostalCode, ShipCountry, OrderDate,
LinkedOrder_Details, LinkedCustomer, LinkedEmployee, LinkedShipper)
VALUES ('VINET', 5, 3, 'Paul Henriot', '59 rue de l'Abbaye', 'Reims',
'51100', 'France', '07/04/1996', 'Order_Details#TEMP', 'Customers#TEMP',
'Employees#TEMP', 'Shippers#TEMP')
```

Code Example

Below is the complete code to create the new Order:

```
Connection conn = DriverManager.getConnection
("jdbc:odata:URL=http://services.odata.org/Northwind/Northwind.svc;");
Statement stat = conn.createStatement();
stat.executeUpdate("INSERT INTO Customers#TEMP (CustomerID, CompanyName,
ContactName, ContactTitle, Address, City, PostalCode, Country, Phone,
Fax) VALUES ('VINET', 'Vins et alcools Chevalier', 'Paul Henriot',
'Accounting Manager', '59 rue de l'Abbaye', 'Reims', '51100', 'France',
'26.47.15.10', '26.47.15.11')");
stat.executeUpdate("INSERT INTO Order_Details#TEMP (ProductID,
UnitPrice, Quantity, Discount) VALUES (72, 34.80, 5, 0)");
stat.executeUpdate("INSERT INTO Order_Details#TEMP (ProductID,
UnitPrice, Quantity, Discount) VALUES (42, 9.80, 10, 0)");
stat.executeUpdate("INSERT INTO Employees#TEMP (EmployeeID) VALUES
(5)");
stat.executeUpdate("INSERT INTO Shippers#TEMP (ShipperID) VALUES (3)");
stat.executeUpdate("INSERT INTO Orders (CustomerID, EmployeeID, ShipVia,
ShipName, ShipAddress, ShipCity, ShipPostalCode, ShipCountry, OrderDate,
LinkedOrder_Details, LinkedCustomer, LinkedEmployee, LinkedShipper)
VALUES ('VINET', 5, 3, 'Paul Henriot', '59 rue de l'Abbaye', 'Reims',
'51100', 'France', '07/04/1996', 'Order_Details#TEMP', 'Customers#TEMP',
```



```
'Employees#TEMP', 'Shippers#TEMP'))");  
stat.close();
```

Views

Modeling Navigation Properties

By default, the adapter models [Navigation Properties](#) as separate views. The views are named in the format *ParentTable_NavigationProperty*. You can disable this behavior with [NavigationPropertiesAsViews](#).

Querying Navigation Properties

For an example of working with a navigation property as a view, consider the Northwind sample service from odata.org. In this service, the Categories entity set has a Products navigation property. The OData Adapter will display a view called Categories_Products for this service. Retrieving data from Categories_Products will display all of the Products associated with a given Category. The Categories_Products view has a primary key made up of the Id of the parent entity and the Id of the related entity.

Support for navigation properties is limited in some OData services. See [NavigationPropertiesAsViews](#) and [SupportsExpand](#) for more information on API restrictions when querying navigation properties.

Navigation Properties

In OData, a navigation property is a property on an entity that is itself either a single entity or list of entities.

A single-entity navigation property signifies a one-to-one relationship; for example, an OData service might allow a Product to have only one Category. In this service, Category might be a navigation property on Products.

An entity set navigation property signifies a one-to-many relationship; for example, many Products can belong in the same Category. In this service, Products might be a navigation property on Categories.

Working with Navigation Properties Relationally

Navigation properties in OData link related entities. Similarly, in a relational database, a foreign key serves to link tables. For example, a Product record might have a CategoryId column, which uniquely identifies what Category the Product belongs to. However, there is no requirement in OData that an entity must contain a foreign key reference to a related entity. This means sometimes you will get navigation properties without having a foreign key reference to that entity on the parent or back to the parent from the related entity. In cases without a foreign key reference, the navigation property's existence is the only thing that can be used to identify a relationship between the two entities.

Select

[NavigationPropertiesAsViews](#) is useful for accessing data in OData services that lack foreign key references. Likewise, it can be used to retrieve related entities that do not exist by themselves such as LineItems on an Invoice. See [Views](#) for more information on querying navigation properties.

Insert

The adapter supports OData deep inserts. See [Tables](#) for more information on specifying navigation properties when you create an entity.

Stored Procedures

Stored procedures are function-like interfaces that extend the functionality of the adapter beyond simple SELECT/INSERT/UPDATE/DELETE operations with OData.

Stored procedures accept a list of parameters, perform their intended function, and then return, if applicable, any relevant response data from OData, along with an indication of whether the procedure succeeded or failed.

OData Adapter Stored Procedures

Name	Description
CreateAssociation	Creates an association between two entities based on a navigation property.

GetOAuthAccessToken	Gets the auth token used to authenticate to the service.
GetOAuthAuthorizationUrl	Gets an authorization URL from the data source. The authorization URL can be used to generate a verifier required to obtain the OAuth token.
GetSAPCSRFToken	Returns a CSRF token and cookie for authentication to SAP.
ListAssociations	Lists associations for a given table and navigation property.
ListNavigationProperties	Lists navigation properties for a given table and the tables they are associated with. Navigation properties are used by the Association stored procedures.
RefreshOAuthAccessToken	Obtains an updated OAuthAccessToken if passed a token to refresh.
RemoveAssociation	Removes an association between two entities based on a navigation property.
Search	Searches OData using the given URL.

CreateAssociation

Creates an association between two entities based on a navigation property.

Input

Name	Type	Description
FromId	<i>String</i>	The Id of the entity you are creating an associations for.
UrlId	<i>String</i>	An alternative to specifying the FromId. This is the complete url to the resource for which the association is being created. It is required to be specified in the case that the navigation property is an abstract, or to

		specify more specific child types where the navigation property entity type is used as a basetype.
FromTable	<i>String</i>	The table where the entity comes from that you are creating an association for. For example, if the FromId was from a table called Customers, set this parameter to: Customers.
ToNavigationProperty	<i>String</i>	The navigation property you are creating an association on. It can be obtained from ListNavigationProperties.
Told	<i>String</i>	The id of the navigation entity. This will come from the table associated with the navigation property.
ToUrlId	<i>String</i>	An alternative to specifying the Told. This is the complete url to the resource to be associated. It is required to be specified in the case that the navigation property is an abstract, or to specify more specific child types where the navigation property entity type is used as a basetype.
HttpMethod	<i>String</i>	An override for the http method to use when creating the association in case the OData source being used does not follow the specifications.

GetOAuthAccessToken

Gets the auth token used to authenticate to the service.

Input

Name	Type	Description
AuthMode	<i>String</i>	The type of authentication you are attempting. Use App for a Windows application, or Web for Web-based applications. The default value is <i>APP</i> .

Verifier	<i>String</i>	A verifier returned by the service that must be input to return the access token. Needed only when using the Web auth mode. Obtained by navigating to the URL returned in <code>GetOAuthAuthorizationUrl</code> .
CallbackUrl	<i>String</i>	The URL the user will be redirected to after authorizing your application.
Other_Options	<i>String</i>	Other options to control behavior of OAuth.
Cert	<i>String</i>	Path for a personal certificate .pfx file. Only available for OAuth 1.0.
Cert_Password	<i>String</i>	Personal certificate password. Only available for OAuth 1.0.
AuthToken	<i>String</i>	The request token returned by navigating to the URL returned by <code>OAuthGetUserAuthorizationURL</code> . Available only for OAuth 1.0.
AuthKey	<i>String</i>	The request secret key returned by <code>OAuthGetUserAuthorizationURL</code> . Available only for OAuth 1.0.
Sign_Method	<i>String</i>	<p>The signature method used to calculate the signature for OAuth 1.0.</p> <p>The allowed values are <i>HMAC-SHA1</i>, <i>PLAINTEXT</i>.</p> <p>The default value is <i>HMAC-SHA1</i>.</p>
GrantType	<i>String</i>	<p>Authorization grant type. Only available for OAuth 2.0.</p> <p>The allowed values are <i>CODE</i>, <i>PASSWORD</i>, <i>CLIENT</i>, <i>REFRESH</i>.</p>
Post_Data	<i>String</i>	The post data to submit, if any.
OAuthParam:*	<i>String</i>	Other parameters may be defined in the format 'param:paramname'.

Result Set Columns

Name	Type	Description
OAuthAccessToken	<i>String</i>	The OAuth access token.
OAuthAccessTokenSecret	<i>String</i>	The OAuth access token secret.
*	<i>String</i>	Other outputs that may be returned by the data source.

GetOAuthAuthorizationUrl

Gets an authorization URL from the data source. The authorization URL can be used to generate a verifier required to obtain the OAuth token.

Input

Name	Type	Description
CallbackURL	<i>String</i>	The URL the user will be redirected to after authorizing your application.
Other_Options	<i>String</i>	Other options to control the behavior of OAuth.
Cert	<i>String</i>	Path for a personal certificate .pfx file. Only available for OAuth 1.0.
Cert_Password	<i>String</i>	Personal certificate password. Only available for OAuth 1.0.
SignMethod	<i>String</i>	The signature method used to calculate the signature for OAuth 1.0.

		<p>The allowed values are <i>HMAC-SHA1</i>, <i>HMAC-SHA256</i>, <i>RSA-SHA1</i>, <i>PLAINTEXT</i>.</p> <p>The default value is <i>HMAC-SHA1</i>.</p>
OAuthParam:*	<i>String</i>	Other parameters may be defined in the format 'param:paramname'.

Result Set Columns

Name	Type	Description
AuthToken	<i>String</i>	The authorization token, passed into the GetOAuthAccessToken stored procedure.
AuthKey	<i>String</i>	The authorization secret token, passed into the GetOAuthAccessToken stored procedure.
URL	<i>String</i>	The authorization URL. This URL will need to be navigated to so that the user can authorize your application to have access. A verifier may be returned when the CallbackURL is redirected to, which will be used in GetOAuthAccessToken.
*	<i>String</i>	Other outputs that may be returned by the data source.

GetSAPCSRFToken

Returns a CSRF token and cookie for authentication to SAP.

Input

Name	Type	Description
------	------	-------------

URL	<i>String</i>	The base URL of the SAP OData service.
TokenHeader	<i>String</i>	The name of the HTTP header for the SAP CSRF token. The default value is <i>x-csrf-token</i> .

Result Set Columns

Name	Type	Description
Success	<i>String</i>	Whether or not the request was successful.
CSRFToken	<i>String</i>	The Cross-Site Request Forgery token to be set in the header for subsequent requests.
Cookie	<i>String</i>	The SAP cookie to be set in the header for subsequent requests.

ListAssociations

Lists associations for a given table and navigation property.

Input

Name	Type	Description
FromId	<i>String</i>	The Id of the entity you are listing associations for.
UrlId	<i>String</i>	An alternative to specifying the FromId. This is the complete url to the resource you are listing the

		associations for. It is required to be specified in the case that the navigation property is an abstract, or to specify more specific child types where the navigation property entity type is used as a basetype.
FromTable	<i>String</i>	The table where the entity comes from that you are listing entities for. For example, if the FromId was from a table called Customers, set this parameter to: Customers.
NavigationProperty	<i>String</i>	The navigation property you are listing associations for. It can be obtained from ListNavigationProperties.

Result Set Columns

Name	Type	Description
Uri	<i>String</i>	The linked url.

ListNavigationProperties

Lists navigation properties for a given table and the tables they are associated with. Navigation properties are used by the Association stored procedures.

Input

Name	Type	Description
TableName	<i>String</i>	The name of the table to list navigation properties for.

Result Set Columns

Name	Type	Description
Name	<i>String</i>	The name of the navigation property.
AssociatedTable	<i>String</i>	The table the navigation property is associated with.

RefreshOAuthAccessToken

Obtains an updated OAuthAccessToken if passed a token to refresh.

Input

Name	Type	Description
OAuthRefreshToken	<i>String</i>	The refresh token returned from the original authorization code exchange.

Result Set Columns

Name	Type	Description
OAuthAccessToken	<i>String</i>	The new OAuthAccessToken returned from the service.
OAuthAccessTokenSecret	<i>String</i>	The new OAuthAccessTokenSecret returned from the service.
OAuthRefreshToken	<i>String</i>	A token that may be used to obtain a new access

		token.
ExpiresIn	<i>String</i>	The remaining lifetime on the access token.

RemoveAssociation

Removes an association between two entities based on a navigation property.

Input

Name	Type	Description
FromId	<i>String</i>	The Id of the entity you are removing an associations for.
UrlId	<i>String</i>	An alternative to specifying the FromId. This is the complete url to the resource you are removing an associations for. It is required to be specified in the case that the navigation property is an abstract, or to specify more specific child types where the navigation property entity type is used as a basetype.
FromTable	<i>String</i>	The table where the entity comes from that you are removing an association for. For example, if the FromId was from a table called Customers, set this parameter to: Customers.
ToNavigationProperty	<i>String</i>	The navigation property you are removing an association on. It can be obtained from ListNavigationProperties.
Told	<i>String</i>	The id of the navigation entity. This will come from the table associated with the navigation property.
ToUrlId	<i>String</i>	An alternative to specifying the Told. This is the complete url to the resource to be associated. It is

required to be specified in the case that the navigation property is an abstract, or to specify more specific child types where the navigation property entity type is used as a basetype.

Search

Searches OData using the given URL.

Input

Name	Type	Description
Url	<i>String</i>	Full URL to use while searching OData.

Result Set Columns

Name	Type	Description
*	<i>String</i>	Output will vary for each entity.

Data Type Mapping

Data Type Mappings

The adapter maps types from the data source to the corresponding data type available in the schema. The table below documents these mappings.

OData V2	OData V3	OData V4	CData Schema
Edm.Binary	Edm.Binary	Edm.Binary	binary
Edm.Boolean	Edm.Boolean	Edm.Boolean	bool
Edm.DateTime	Edm.DateTime	Edm.DateTimeOffset	datetime
Edm.Decimal	Edm.Decimal	Edm.Decimal	decimal
Edm.Double	Edm.Double	Edm.Double	double
Edm.Guid	Edm.Guid	Edm.Guid	guid
Edm.Int16	Edm.Int16	Edm.Int16	int
Edm.Int32	Edm.Int32	Edm.Int32	int
Edm.Int64	Edm.Int64	Edm.Int64	bigint
Edm.String	Edm.String	Edm.String	string
Edm.Time	Edm.Time	Edm.TimeOfDay	time

Connection String Options

The connection string properties are the various options that can be used to establish a connection. This section provides a complete list of the options you can configure in the connection string for this provider. Click the links for further details.

For more information on establishing a connection, see [Basic Tab](#).

Authentication

Property	Description
URL	URL to the Organization root or the OData services file. For example,

	http://MySite/MyOrganization.
AuthScheme	The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, NEGOTIATE, or SHAREPOINTONLINE.
User	The OData user account used to authenticate.
Password	The password used to authenticate the user.
FeedURL	URL to the OData entity set. For example, http://MySite/MyOrganization/EntitySet.
SharePointUseSSO	Whether or not to use single sign-on (SSO) to authenticate to SharePoint Online.

Azure Authentication

Property	Description
AzureADTenant	The Azure Active Directory tenant to authenticate against (only used with Azure AD OAuth).
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureResource	The Azure Active resource to authenticate to (used during Azure OAuth exchange).

SSO

Property	Description
SharePointSSODomain	The domain of the user when using single sign-on (SSO).

OAuth

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client Id assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	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://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.

OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.
AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

Kerberos

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.

KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

SSL

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.
SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

Firewall

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

Proxy

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.
ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

Logging

Property	Description
LogModules	Core modules to be included in the log file.

Schema

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

Miscellaneous

Property	Description
ContinueOnError	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.
CustomHeaders	Other headers as determined by the user (optional).
CustomUrlParams	The custom query string to be included in the request.
DataFormat	The data format to retrieve data in. Select either ATOM or JSON.
EnableAtomicBatchOperations	Whether or not the CData ADO.NET Provider for OData should use atomic batch operations.
IncludeNavigationParentColumns	Indicates if navigation parent columns should be included on navigation views.
IncludeReferenceColumn	Adds a input only ParentReference column for bulk inserts to properly associate children during a deep insert with the same parent.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
NavigationPropertiesAsViews	A boolean indicating navigation properties should be promoted to full views.
ODataVersion	The version of OData to use. By default the provider will attempt to autodetect the version.
Other	These hidden properties are used only in specific use cases.

Pagesize	The maximum number of results to return per page from OData.
Readonly	You can use this property to enforce read-only access to OData from the provider.
ServerTimeZone	The timezone by which the server's Edm.DateTime values are represented. The value of this property will affect how Edm.DateTime filters and results are converted between the server and the client machine.
StoredProceduresAsViews	A boolean indicating if we should list stored procedures which return a collection of entities as views.
SupportsExpand	Whether you need to specify the base entity's key to query navigation property views.
SupportsFormulas	A boolean indicating if the odata service supports server side formulas.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseClientSidePaging	Whether or not the CData ADO.NET Provider for OData should use client side paging.
UseEtags	Whether or not the OData source uses Etags.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.
UseSimpleNames	Boolean determining if simple names should be used for tables and columns.

Authentication

This section provides a complete list of the Authentication properties you can configure in the connection string for this provider.

Property	Description
URL	URL to the Organization root or the OData services file. For example, http://MySite/MyOrganization .
AuthScheme	The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, NEGOTIATE, or SHAREPOINTONLINE.
User	The OData user account used to authenticate.
Password	The password used to authenticate the user.
FeedURL	URL to the OData entity set. For example, http://MySite/MyOrganization/EntitySet .
SharePointUseSSO	Whether or not to use single sign-on (SSO) to authenticate to SharePoint Online.

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>.

AuthScheme

The scheme used for authentication. Accepted entries are NTLM, BASIC, DIGEST, NONE, NEGOTIATE, or SHAREPOINTONLINE.

Possible Values

None, AzureAD, Basic, Digest, Negotiate, NTLM, OAuth, 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.

- None: No authentication for this service.
- AzureAD: Set this to perform Azure Active Directory OAuth 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.
- NTLM: Set this to use your Windows credentials for authentication.
- OAuth: Set this to establish an OAuth connection.
- SharePointOnline: Set this to use SharePoint Online authentication.

User

The OData user account used to authenticate.

Data Type

string

Default Value

""

Remarks

Together with [Password](#), this field is used to authenticate against the OData server.

Password

The password used to authenticate the user.

Data Type

string

Default Value

""

Remarks

The [User](#) and [Password](#) are together used to authenticate with the server.

FeedURL

URL to the OData entity set. For example, <http://MySite/MyOrganization/EntitySet>.

Data Type

string

Default Value

""

Remarks

URL to the OData entity set. For example, <http://MySite/MyOrganization/EntitySet>. You can use this property when the OData service does not have a root document.

SharePointUseSSO

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.

[SharePointSSODomain](#) 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 and AuthScheme is set to SHAREPOINTONLINE. It is not available for OAuth connections to SharePoint.

Azure Authentication

This section provides a complete list of the Azure Authentication properties you can configure in the connection string for this provider.

Property	Description
AzureADTenant	The Azure Active Directory tenant to authenticate against (only used with Azure AD OAuth).
AzureTenant	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
AzureResource	The Azure Active resource to authenticate to (used during Azure OAuth exchange).

AzureADTenant

The Azure Active Directory tenant to authenticate against (only used with Azure AD OAuth).

Data Type

string

Default Value

""

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.

AzureTenant

The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.

Data Type

string

Default Value

""

Remarks

The Microsoft Online tenant being used to access data. For instance, contoso.onmicrosoft.com. Alternatively, specify the tenant Id. This value is the directory Id in the Azure Portal > Azure Active Directory > Properties.

Typically it is not necessary to specify the Tenant. This can be automatically determined by Microsoft when using the [OAuthGrantType](#) set to CODE (default). However, it may fail in the case that the user belongs to multiple tenants. For instance, if an Admin of domain A invites a user of domain B to be a guest user. The user will now belong to both tenants. It is a good practice to specify the Tenant, although in general things should normally work without having to specify it.

The [AzureTenant](#) is required when setting [OAuthGrantType](#) to CLIENT. When using client credentials, there is no user context. The credentials are taken from the context of the app itself. While Microsoft still allows client credentials to be obtained without specifying which Tenant, it has a much lower probability of picking the specific tenant you want to work with. For this reason, we require [AzureTenant](#) to be explicitly stated for all client credentials connections to ensure you get credentials that are applicable for the domain you intend to connect to.

AzureResource

The Azure Active resource to authenticate to (used during Azure OAuth exchange).

Data Type

string

Default Value

""

Remarks

The resource must be specified if using Azure OAuth. It should be set to the App Id URI of the web API (secured resource).

SSO

This section provides a complete list of the SSO properties you can configure in the connection string for this provider.

Property	Description
SharePointSSODomain	The domain of the user when using single sign-on (SSO).

SharePointSSODomain

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 ([SharePointUseSSO](#) 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.

OAuth

This section provides a complete list of the OAuth properties you can configure in the connection string for this provider.

Property	Description
InitiateOAuth	Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.
OAuthVersion	The version of OAuth being used.
OAuthClientId	The client Id assigned when you register your application with an OAuth authorization server.
OAuthClientSecret	The client secret assigned when you register your application with an OAuth authorization server.
OAuthAccessToken	The access token for connecting using OAuth.
OAuthAccessTokenSecret	The OAuth access token secret for connecting using OAuth.
OAuthSettingsLocation	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://.
CallbackURL	The OAuth callback URL to return to when authenticating. This value must match the callback URL you specify in your app settings.
OAuthGrantType	The grant type for the OAuth flow.
OAuthIncludeCallbackURL	Whether to include the callback URL in an access token request.
OAuthAuthorizationURL	The authorization URL for the OAuth service.
OAuthAccessTokenURL	The URL to retrieve the OAuth access token from.
OAuthRefreshTokenURL	The URL to refresh the OAuth token from.
OAuthRequestTokenURL	The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.

OAuthVerifier	The verifier code returned from the OAuth authorization URL.
AuthToken	The authentication token used to request and obtain the OAuth Access Token.
AuthKey	The authentication secret used to request and obtain the OAuth Access Token.
OAuthParams	A comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.
OAuthRefreshToken	The OAuth refresh token for the corresponding OAuth access token.
OAuthExpiresIn	The lifetime in seconds of the OAuth AccessToken.
OAuthTokenTimestamp	The Unix epoch timestamp in milliseconds when the current Access Token was created.

InitiateOAuth

Set this property to initiate the process to obtain or refresh the OAuth access token when you connect.

Possible Values

OFF, GETANDREFRESH, REFRESH

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.

OAuthVersion

The version of OAuth being used.

Possible Values

1.0, 2.0

Data Type

string

Default Value

"2.0"

Remarks

The version of OAuth being used. The following options are available: 1.0,2.0

OAuthClientId

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](#).

OAuthClientSecret

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.

OAuthAccessToken

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 user name and password. The access token protects your credentials by keeping them on the server.

OAuthAccessTokenSecret

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.

OAuthSettingsLocation

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\\OData 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.

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.

If left unspecified, the default location is "%APPDATA%\\CData\\OData Data Provider\\OAuthSettings.txt" with **%APPDATA%** being set to the user's configuration directory:

Platform	%APPDATA%
Windows	The value of the APPDATA environment variable
Mac	~/Library/Application Support
Linux	~/.config

CallbackURL

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.

OAuthGrantType

The grant type for the OAuth flow.

Possible Values

CODE, CLIENT, PASSWORD

Data Type

string

Default Value

"CODE"

Remarks

The following options are available: CODE,CLIENT,PASSWORD

OAuthIncludeCallbackURL

Whether to include the callback URL in an access token request.

Data Type

bool

Default Value

true

Remarks

This defaults to true since standards-compliant OAuth services will ignore the redirect_uri parameter for grant types like CLIENT or PASSWORD that do not require it.

This option should only be enabled for OAuth services that report errors when redirect_uri is included.

OAuthAuthorizationURL

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.

OAuthAccessTokenURL

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.

OAuthRefreshTokenURL

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.

OAuthRequestTokenURL

The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0.

Data Type

string

Default Value

""

Remarks

The URL the service provides to retrieve request tokens from. This is required in OAuth 1.0. In OAuth 1.0, this is the URL where the app makes a request for the request token.

OAuthVerifier

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 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.

AuthToken

The authentication token used to request and obtain the OAuth Access Token.

Data Type

string

Default Value

""

Remarks

This property is required only when performing headless authentication in OAuth 1.0. It can be obtained from the GetOAuthAuthorizationUrl stored procedure.

It can be supplied alongside the [AuthKey](#) in the GetOAuthAccessToken stored procedure to obtain the [OAuthAccessToken](#).

AuthKey

The authentication secret used to request and obtain the OAuth Access Token.

Data Type

string

Default Value

""

Remarks

This property is required only when performing headless authentication in OAuth 1.0. It can be obtained from the GetOAuthAuthorizationUrl stored procedure.

It can be supplied alongside the [AuthToken](#) in the GetOAuthAccessToken stored procedure to obtain the [OAuthAccessToken](#).

OAuthParams

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.

OAuthRefreshToken

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.

OAuthExpiresIn

The lifetime in seconds of the OAuth AccessToken.

Data Type

string

Default Value

""

Remarks

Pair with OAuthTokenTimestamp to determine when the AccessToken will expire.

OAuthTokenTimestamp

The Unix epoch timestamp in milliseconds when the current Access Token was created.

Data Type

string

Default Value

""

Remarks

Pair with OAuthExpiresIn to determine when the AccessToken will expire.

Kerberos

This section provides a complete list of the Kerberos properties you can configure in the connection string for this provider.

Property	Description
KerberosKDC	The Kerberos Key Distribution Center (KDC) service used to authenticate the user.
KerberosRealm	The Kerberos Realm used to authenticate the user.
KerberosSPN	The service principal name (SPN) for the Kerberos Domain Controller.
KerberosKeytabFile	The Keytab file containing your pairs of Kerberos principals and encrypted keys.
KerberosServiceRealm	The Kerberos realm of the service.
KerberosServiceKDC	The Kerberos KDC of the service.
KerberosTicketCache	The full file path to an MIT Kerberos credential cache file.

KerberosKDC

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 SPNEGO or Windows Authentication. The adapter will request session tickets and temporary session keys from the Kerberos KDC service. The Kerberos 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:

- **KRB5 Config File (krb5.ini/krb5.conf):** If the KRB5_CONFIG environment variable is set and the file exists, the adapter will obtain the KDC from the specified file. Otherwise, it will attempt to read from the default MIT location based on the OS: `C:\ProgramData\MIT\Kerberos5\krb5.ini` (Windows) or `/etc/krb5.conf` (Linux).
- **Java System Properties:** Using the system properties `java.security.krb5.realm` and `java.security.krb5.kdc`.
- **Domain Name and Host:** If the Kerberos Realm and Kerberos KDC could not be inferred from another location, the adapter will infer them from the configured domain name and host.

Note: Windows authentication is supported in JRE 1.6 and above only.

KerberosRealm

The Kerberos Realm used to authenticate the user.

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:

- **KRB5 Config File (krb5.ini/krb5.conf):** If the KRB5_CONFIG environment variable is set and the file exists, the adapter will obtain the default realm from the specified file. Otherwise, it will attempt to read from the default MIT location based on the OS: `C:\ProgramData\MIT\Kerberos5\krb5.ini` (Windows) or `/etc/krb5.conf` (Linux)
- **Java System Properties:** Using the system properties `java.security.krb5.realm` and `java.security.krb5.kdc`.

- **Domain Name and Host:** If the Kerberos Realm and Kerberos KDC could not be inferred from another location, the adapter will infer them from the user-configured domain name and host. This might work in some Windows environments.

Note: Kerberos-based authentication is supported in JRE 1.6 and above only.

KerberosSPN

The service principal name (SPN) for the Kerberos Domain Controller.

Data Type

string

Default Value

""

Remarks

If the SPN on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, use this property to set the SPN.

KerberosKeytabFile

The Keytab file containing your pairs of Kerberos principals and encrypted keys.

Data Type

string

Default Value

""

Remarks

The Keytab file containing your pairs of Kerberos principals and encrypted keys.

KerberosServiceRealm

The Kerberos realm of the service.

Data Type

string

Default Value

""

Remarks

The KerberosServiceRealm is the specify the service Kerberos realm when using cross-realm Kerberos authentication.

In most cases, a single realm and KDC machine are used to perform the Kerberos authentication and this property is not required.

This property is available for complex setups where a different realm and KDC machine are used to obtain an authentication ticket (AS request) and a service ticket (TGS request).

KerberosServiceKDC

The Kerberos KDC of the service.

Data Type

string

Default Value

""

Remarks

The KerberosServiceKDC is used to specify the service Kerberos KDC when using cross-realm Kerberos authentication.

In most cases, a single realm and KDC machine are used to perform the Kerberos authentication and this property is not required.

This property is available for complex setups where a different realm and KDC machine are used to obtain an authentication ticket (AS request) and a service ticket (TGS request).

KerberosTicketCache

The full file path to an MIT Kerberos credential cache file.

Data Type

string

Default Value

""

Remarks

This property can be set if you wish to use a credential cache file that was created using the MIT Kerberos Ticket Manager or kinit command.

SSL

This section provides a complete list of the SSL properties you can configure in the connection string for this provider.

Property	Description
SSLClientCert	The TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).
SSLClientCertType	The type of key store containing the TLS/SSL client certificate.
SSLClientCertPassword	The password for the TLS/SSL client certificate.

SSLClientCertSubject	The subject of the TLS/SSL client certificate.
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

SSLClientCert

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. See [SSLClientCertSubject](#) for more information.

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 (for example, PKCS12 certificate store).

SSLClientCertType

The type of key store containing the TLS/SSL client certificate.

Possible Values

USER, MACHINE, PFXFILE, PFXBLOB, JKSFIL, JKSBLOB, PEMKEY_FILE, PEMKEY_BLOB, PUBLIC_KEY_FILE, PUBLIC_KEY_BLOB, SSHPUBLIC_KEY_FILE, SSHPUBLIC_KEY_BLOB, P7BFILE, PPKFILE, XMLFILE, XMLBLOB

Data Type

string

Default Value

"USER"

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 that this store type is not available in Java.
MACHINE	For Windows, this specifies that the certificate store is a machine

	store. Note that 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 that this store type is only available in Java.
JKSBLOB	The certificate store is a string (base-64-encoded) representing a certificate store in JKS format. Note that 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 PuTTY Private Key (PPK).

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.

SSLClientCertPassword

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 to open the certificate store.

SSLClientCertSubject

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 example, "CN=www.server.com, OU=test, C=US, E=support@company.com". The common fields and their meanings are shown 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.

SSLServerCert

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 is rejected.

This property can take the following forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIICHTCCAe4CAQAwDQYJKoZIhvd.....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)	34e929226ae0819f2ec14b4a3d904f801c
The SHA1 Thumbprint (hex values can also be either space or colon separated)	bb150d

If not specified, any certificate trusted by the machine is accepted.

Certificates are validated as trusted by the machine based on the System's trust store. The trust store used is the 'javax.net.ssl.trustStore' value specified for the system. If no value is specified for this property, Java's default trust store is used (for example, JAVA_HOME\lib\security\cacerts).

Use '*' to signify to accept all certificates. Note that this is not recommended due to security concerns.

Firewall

This section provides a complete list of the Firewall properties you can configure in the connection string for this provider.

Property	Description
FirewallType	The protocol used by a proxy-based firewall.
FirewallServer	The name or IP address of a proxy-based firewall.
FirewallPort	The TCP port for a proxy-based firewall.
FirewallUser	The user name to use to authenticate with a proxy-based firewall.
FirewallPassword	A password used to authenticate to a proxy-based firewall.

FirewallType

The protocol used by a proxy-based firewall.

Possible Values

NONE, TUNNEL, SOCKS4, SOCKS5

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](#).

FirewallServer

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.

FirewallPort

The TCP port for a proxy-based firewall.

Data Type

int

Default Value

0

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](#).

FirewallUser

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](#).

FirewallPassword

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](#).

Proxy

This section provides a complete list of the Proxy properties you can configure in the connection string for this provider.

Property	Description
ProxyAutoDetect	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.
ProxyServer	The hostname or IP address of a proxy to route HTTP traffic through.
ProxyPort	The TCP port the ProxyServer proxy is running on.
ProxyAuthScheme	The authentication type to use to authenticate to the ProxyServer proxy.
ProxyUser	A user name to be used to authenticate to the ProxyServer proxy.

ProxyPassword	A password to be used to authenticate to the ProxyServer proxy.
ProxySSLType	The SSL type to use when connecting to the ProxyServer proxy.
ProxyExceptions	A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

ProxyAutoDetect

This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.

Data Type

bool

Default Value

true

Remarks

This takes precedence over other proxy settings, so you'll need to set ProxyAutoDetect to FALSE in order use custom proxy settings.

NOTE: When this property is set to True, the proxy used is determined as follows:

- A search from the JVM properties (**http.proxy**, **https.proxy**, **socksProxy**, etc.) is performed.
- In the case that the JVM properties don't exist, a search from **java.home/lib/net.properties** is performed.
- In the case that java.net.useSystemProxies is set to True, a search from **the SystemProxy** is performed.
- In Windows only, an attempt is made to retrieve these properties from the **Internet Options** in the **registry**.

To connect to an HTTP proxy, see [ProxyServer](#). For other proxies, such as SOCKS or tunneling, see [FirewallType](#).

ProxyServer

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.

ProxyPort

The TCP port the ProxyServer proxy is running on.

Data Type

int

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](#).

ProxyAuthScheme

The authentication type to use to authenticate to the ProxyServer proxy.

Possible Values

BASIC, DIGEST, NONE, NEGOTIATE, NTLM, PROPRIETARY

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](#).

ProxyUser

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 user name of a user recognized by the HTTP proxy. If you are using Windows or Kerberos authentication, set this property to a user name in one of the following formats:

```
user@domain  
domain\user
```

ProxyPassword

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.

ProxySSLType

The SSL type to use when connecting to the [ProxyServer](#) proxy.

Possible Values

AUTO, ALWAYS, NEVER, TUNNEL

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.

ProxyExceptions

A semicolon separated list of destination hostnames or IPs that are exempt from connecting through the ProxyServer .

Data Type

string

Default Value

""

Remarks

The [ProxyServer](#) is used for all addresses, except for addresses defined in this property. Use semicolons to separate entries.

Note that the adapter uses the system proxy settings by default, without further configuration needed; if you want to explicitly configure proxy exceptions for this connection, you need to set [ProxyAutoDetect](#) = false, and configure [ProxyServer](#) and [ProxyPort](#). To authenticate, set [ProxyAuthScheme](#) and set [ProxyUser](#) and [ProxyPassword](#), if needed.

Logging

This section provides a complete list of the Logging properties you can configure in the connection string for this provider.

Property	Description
LogModules	Core modules to be included in the log file.

LogModules

Core modules to be included in the log file.

Data Type

string

Default Value

""

Remarks

Only the modules specified (separated by ';') will be included in the log file. By default all modules are included.

See the [Logging](#) page for an overview.

Schema

This section provides a complete list of the Schema properties you can configure in the connection string for this provider.

Property	Description
Location	A path to the directory that contains the schema files defining tables, views, and stored procedures.

Location

A path to the directory that contains the schema files defining tables, views, and stored procedures.

Data Type

string

Default Value

"%APPDATA%\\CData\\OData Data Provider\\Schema"

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 folder location can be a relative path from the location of the executable. The Location property is only needed if you want to customize definitions (for example, change a column name, ignore a column, and so on) or extend the data model with new tables, views, or stored procedures.

If left unspecified, the default location is "%APPDATA%\\CData\\OData Data Provider\\Schema" with **%APPDATA%** being set to the user's configuration directory:

Platform	%APPDATA%
Windows	The value of the APPDATA environment variable
Mac	~/Library/Application Support
Linux	~/.config

Miscellaneous

This section provides a complete list of the Miscellaneous properties you can configure in the connection string for this provider.

Property	Description
ContinueOnError	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.
CustomHeaders	Other headers as determined by the user (optional).
CustomUrlParams	The custom query string to be included in the request.
DataFormat	The data format to retrieve data in. Select either ATOM or JSON.
EnableAtomicBatchOperations	Whether or not the CData ADO.NET Provider for OData should use atomic batch operations.
IncludeNavigationParentColumns	Indicates if navigation parent columns should be included on navigation views.
IncludeReferenceColumn	Adds a input only ParentReference column for bulk inserts to properly associate children during a deep insert with the same parent.
MaxRows	Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.
NavigationPropertiesAsViews	A boolean indicating navigation properties should be promoted to full views.
ODataVersion	The version of OData to use. By default the provider will attempt to autodetect the version.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from OData.

Readonly	You can use this property to enforce read-only access to OData from the provider.
ServerTimeZone	The timezone by which the server's Edm.DateTime values are represented. The value of this property will affect how Edm.DateTime filters and results are converted between the server and the client machine.
StoredProceduresAsViews	A boolean indicating if we should list stored procedures which return a collection of entities as views.
SupportsExpand	Whether you need to specify the base entity's key to query navigation property views.
SupportsFormulas	A boolean indicating if the odata service supports server side formulas.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UseClientSidePaging	Whether or not the CData ADO.NET Provider for OData should use client side paging.
UseEtags	Whether or not the OData source uses Etags.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.
UseSimpleNames	Boolean determining if simple names should be used for tables and columns.

ContinueOnError

Whether or not to continue after encountering an error on a batch request.

Data Type

bool

Default Value

true

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.

CustomHeaders

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`, and so on.

The headers must be of the format "header: value" as described in the HTTP specifications. Header lines should be separated by the carriage return and line feed (CRLF) 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.

CustomUrlParams

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`. The values in the query string must be URL encoded.

DataFormat

The data format to retrieve data in. Select either `ATOM` or `JSON`.

Possible Values

AUTO, ATOM, JSON

Data Type

string

Default Value

"AUTO"

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.

EnableAtomicBatchOperations

Whether or not the CData ADO.NET Provider for OData should use atomic batch operations.

Data Type

bool

Default Value

true

Remarks

Whether or not the OData Adapter should use atomic batch operations.

IncludeNavigationParentColumns

Indicates if navigation parent columns should be included on navigation views.

Data Type

bool

Default Value

true

Remarks

When [NavigationPropertiesAsViews](#) is set to true, this property controls if parent columns from the navigation property will be displayed or not on the view. It may be worth displaying them in order to take advantage of being able to filter based on information about the parent.

When set to false, the primary keys of the parent will still be displayed to allow for joining back to the parent, but other other columns will not be.

IncludeReferenceColumn

Adds a input only ParentReference column for bulk inserts to properly associate children during a deep insert with the same parent.

Data Type

bool

Default Value

false

Remarks

Adds a input only ParentReference column for bulk inserts to properly associate children during a deep insert with the same parent.

MaxRows

Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

Data Type

int

Default Value

-1

Remarks

Limits the number of rows returned rows when no aggregation or group by is used in the query. This helps avoid performance issues at design time.

NavigationPropertiesAsViews

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.

ODataVersion

The version of OData to use. By default the provider will attempt to autodetect the version.

Possible Values

AUTO, 2.0, 3.0, 4.0

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

These hidden properties are used 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	Determines 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 OData.

Data Type

int

Default Value

1000

Remarks

The Pagesize property affects the maximum number of results to return per page from OData. Setting a higher value may result in better performance at the cost of additional

memory allocated per page consumed.

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.

ServerTimeZone

The timezone by which the server's Edm.DateTime values are represented. The value of this property will affect how Edm.DateTime filters and results are converted between the server and the client machine.

Data Type

string

Default Value

""

Remarks

By default, Edm.DateTime values in the server will be assumed to be GMT. If the server is known to represent such values in a specific timezone, then the abbreviation of that timezone can be provided here (i.e. EST). From there, the driver will convert any

Edm.DateTime derived filters from the installed machine's local timezone to the one specified for the server. Conversely, similar values returned by the OData server will be converted from the specified timezone to the installed machine's local timezone before being exposed in the result set.

StoredProceduresAsViews

A boolean indicating if we should list stored procedures which return a collection of entities as views.

Data Type

bool

Default Value

true

Remarks

A boolean indicating if we should list stored procedures which return a collection of entities as views.

SupportsExpand

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 entities.

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)
```

SupportsFormulas

A boolean indicating if the odata service supports server side formulas.

Data Type

bool

Default Value

false

Remarks

OData has a number of server side formulas that are built into the specifications. However, many services do not natively support them and will return errors when these formulas are appended to the \$filter parameter. These formulas can be used to make some queries that use them execute much faster. If your OData service supports formulas, change this connection property to true. Otherwise, leave it as false.

Timeout

The value in seconds until the timeout error is thrown, canceling the operation.

Data Type

int

Default Value

60

Remarks

If Timeout = 0, operations do not time out. The operations run until they complete successfully or until they encounter an error condition.

If Timeout expires and the operation is not yet complete, the adapter throws an exception.

UseClientSidePaging

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.

UseEtags

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.

UserDefinedViews

A filepath pointing to the JSON configuration file containing your custom views.

Data Type

string

Default Value

""

Remarks

User Defined Views are defined in a JSON-formatted configuration file called *UserDefinedViews.json*. The adapter automatically detects the views specified in this file.

You can also have multiple view definitions and control them using the UserDefinedViews connection property. When you use this property, only the specified views are seen by the adapter.

This User Defined View configuration file is formatted as follows:

- Each root element defines the name of a view.
- Each root element contains a child element, called **query**, which contains the custom SQL query for the view.

For example:

```
{
  "MyView": {
    "query": "SELECT * FROM Lead WHERE MyColumn = 'value'"
  },
  "MyView2": {
    "query": "SELECT * FROM MyTable WHERE Id IN (1,2,3)"
  }
}
```

Use the UserDefinedViews connection property to specify the location of your JSON configuration file. For example:

```
"UserDefinedViews",
"C:\\Users\\yourusername\\Desktop\\tmp\\UserDefinedViews.json"
```

UseSimpleNames

Boolean determining if simple names should be used for tables and columns.

Data Type

bool

Default Value

false

Remarks

OData 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.

TIBCO Product Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join the TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the [TIBCO® Data Virtualization](#) page.

- **Users**
 - TDV Getting Started Guide
 - TDV User Guide
 - TDV Web UI User Guide
 - TDV Client Interfaces Guide
 - TDV Tutorial Guide
 - TDV Northbay Example
- **Administration**
 - TDV Installation and Upgrade Guide
 - TDV Administration Guide
 - TDV Active Cluster Guide
 - TDV Security Features Guide
- **Data Sources**

TDV Adapter Guides

TDV Data Source Toolkit Guide (Formerly Extensibility Guide)

- **References**

TDV Reference Guide

TDV Application Programming Interface Guide

- **Other**

TDV Business Directory Guide

TDV Discovery Guide

- *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.

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- 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 [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on the website.

Release Version Support

TDV 8.5 is designated as a Long Term Support (LTS) version. Some release versions of TIBCO® Data Virtualization products are selected to be long-term support (LTS) versions. Defect corrections will typically be delivered in a new release version and as hotfixes or service packs to one or more LTS versions. See also

https://docs.tibco.com/pub/tdv/general/LTS/tdv_LTS_releases.htm.

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, visit [TIBCO Community](#).

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, TIBCO logo, TIBCO O logo, ActiveSpaces, Enterprise Messaging Service, Spotfire, TERR, S-PLUS, and S+ are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the

readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2002-2023 Cloud Software Group, Inc All Rights Reserved.