



TIBCO® Data Virtualization

Marklogic Adapter Guide

Version 8.7.0 | October 2023

Contents

Contents	2
MarkLogic Adapter	4
Getting Started	4
Basic Tab	5
Logging	5
Picking the right API	7
Changelog	8
Advanced Features	11
User Defined Views	12
SSL Configuration	14
Firewall and Proxy	15
Query Processing	16
Logging	16
SQL Compliance	19
SELECT Statements	20
Aggregate Functions	23
Projection Functions	24
Predicate Functions	42
SELECT INTO Statements	60
EXECUTE Statements	61
PIVOT and UNPIVOT	61
Data Model	62
Connection String Options	63
Authentication	66
SSL	70
Firewall	76
Proxy	79

Logging	86
Schema	86
Miscellaneous	88
TIBCO Product Documentation and Support Services	92
How to Access TIBCO Documentation	92
How to Contact TIBCO Support	93
Release Version Support	93
How to Join TIBCO Community	94
Legal and Third-Party Notices	95

MarkLogic Adapter

MarkLogic Version Support

The MarkLogic Adapter connects with the MarkLogic Web Services API.

SQL Compliance

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

Getting Started

Connecting to MarkLogic

[Basic Tab](#) shows how to authenticate to MarkLogic 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 MarkLogic Adapter

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

1. Unzip the `tdv.marklogic.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.marklogic/tdv.marklogic.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 MarkLogic
```

Basic Tab

Connecting to MarkLogic

Picking an API

The MarkLogic Adapter offers access to both the REST and ODBC APIs for MarkLogic. Please see [Picking the right API](#) for a discussion on which API is right for your use case.

Connection Details

Regardless of which API you are connecting to, you will need to include the following connection details:

- Server - The server being connected to.
- Database - (Optional) The specific database to connect to.
- Port - (Optional) The specific port to use if the default port for the selected API is not being used default.

Create an ODBC Server

If connecting to the ODBC API, please follow the guide in MarkLogic's official website on how to create an ODBC Server, <https://docs.marklogic.com/guide/admin/odbc>.

Authenticating to MarkLogic

To authenticate to MarkLogic, specify the following connection properties:

- User - The user to authenticate as.
- Password - The password of the user.

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 MarkLogic 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**.

Picking the right API

This section outlines the differences between the ODBC server API and the REST one. The MarkLogic Adapter offers the possibility to connect to either of the APIs using the connection property API. Both APIs model the same data model (catalogs, schemas, views, and columns).

The ODBC Server API

The ODBC version connects with a PostgreSQL front end on the client by means of the PostgreSQL message protocol. Everything is offloaded to the API. This API is more performant than the REST API. By default, the MarkLogic Adapter will use the ODBC server API. SupportEnhancedSQL is not supported when the ODBC API is selected.

The REST API

The REST API uses the Optic API to query documents. Unlike the ODBC server, some operations are not supported server-side. Some of these operations include formula columns without namespaces, group by, and having expressions in some instances. These operations are instead handled on the client side. The property `SupportEnhancedSQL` can be used to control this behaviour.

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.
09/02/2021	7915	General	Added <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	Changed <ul style="list-style-type: none"> Added the KeySeq column to the sys_

foreignkeys table.			
08/06/2021	7888	General	Changed <ul style="list-style-type: none"> Added the new sys_primarykeys system table.
07/23/2021	7874	General	Changed <ul style="list-style-type: none"> Updated the Literal Function Names for relative date/datetime functions. Previously relative date/datetime functions resolved to a different value when used in the projection vs te predicate. Ie: 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. As a replacement for the previous behavior, the relative date/datetime functions in the criteria may have an 'L' appended to them. Ie: 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	Added <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.
04/23/2021	7785	General	Added <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	Changed <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	Added <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 Changed <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. 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
03/03/2021	7733	MarkLogic	Changed <ul style="list-style-type: none"> • Default Integer DisplaySize changed from 16 to 4. • Default VARCHAR Length changed from 255 to 2000.

Advanced Features

This section details a selection of advanced features of the MarkLogic 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 MarkLogic 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 MarkLogic 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 [CData].[main].Customer 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.

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 MarkLogic 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 MarkLogic 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 MarkLogic Query Evaluation component examines SQL queries and returns information indicating what parts of the query the adapter is not capable of executing natively.

The MarkLogic 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 <u>Verbosity</u> 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 <u>Verbosity</u> to 2 will log everything included in <u>Verbosity</u> 1 and additional information about the request. |
| 3 | Setting <u>Verbosity</u> to 3 will additionally log HTTP headers, as well as the body of the request and the response. |
| 4 | Setting <u>Verbosity</u> 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

SELECT Statements

See [SELECT Statements](#) for a syntax reference and examples.

See [Data Model](#) for information on the capabilities of the MarkLogic API.

EXECUTE Statements

Use EXECUTE or EXEC statements to execute stored procedures. See [EXECUTE Statements](#) for a syntax reference and examples.

Names and Quoting

- Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-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 MarkLogic 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> ]
  [ GROUP BY <column_reference> [ , ... ] ]
  [ HAVING <search_condition> ]
  [
    ORDER BY
    <column_reference> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ]
  ]
  [
    LIMIT <expression>
  ]
}

<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> { = | > | < | >= | <= | <> | != | LIKE | NOT LIKE |
IN | NOT IN | IS NULL | IS NOT NULL | AND | OR } [ <expression> ]
  } [ { AND | OR } ... ]

```

Examples

1. Return all columns:

```
SELECT * FROM [CData].[main].Customer
```

2. Rename a column:

```
SELECT "TotalDue" AS MY_TotalDue FROM [CData].[main].Customer
```

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

```
SELECT CAST(AnnualRevenue AS VARCHAR) AS Str_AnnualRevenue FROM  
[CData].[main].Customer
```

4. Search data:

```
SELECT * FROM [CData].[main].Customer WHERE CustomerId = '12345'
```

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

```
SELECT * FROM [CData].[main].Customer WHERE CustomerId = '12345';
```

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

```
SELECT COUNT(*) AS MyCount FROM [CData].[main].Customer
```

7. Return the number of unique items matching the query criteria:

```
SELECT COUNT(DISTINCT TotalDue) FROM [CData].[main].Customer
```

8. Summarize data:

```
SELECT TotalDue, MAX(AnnualRevenue) FROM [CData].[main].Customer  
GROUP BY TotalDue
```

See [Aggregate Functions](#) for details.

9. Sort a result set in ascending order:

```
SELECT Name, TotalDue FROM [CData].[main].Customer ORDER BY
TotalDue ASC
```

Aggregate Functions

Examples of Aggregate Functions

Below are several examples of SQL aggregate functions. You can use these with a GROUP BY clause to aggregate rows based on the specified GROUP BY criterion. This can be a reporting tool.

COUNT

Returns the number of rows matching the query criteria.

```
SELECT COUNT(*) FROM [CData].[main].Customer WHERE CustomerId = '12345'
```

COUNT(DISTINCT)

Returns the number of distinct, non-null field values matching the query criteria.

```
SELECT COUNT(DISTINCT Name) AS DistinctValues FROM [CData].
[main].Customer WHERE CustomerId = '12345'
```

AVG

Returns the average of the column values.

```
SELECT TotalDue, AVG(AnnualRevenue) FROM [CData].[main].Customer WHERE
CustomerId = '12345' GROUP BY TotalDue
```

MIN

Returns the minimum column value.

```
SELECT MIN(AnnualRevenue), TotalDue FROM [CData].[main].Customer WHERE
CustomerId = '12345' GROUP BY TotalDue
```

MAX

Returns the maximum column value.

```
SELECT TotalDue, MAX(AnnualRevenue) FROM [CData].[main].Customer WHERE
CustomerId = '12345' GROUP BY TotalDue
```

SUM

Returns the total sum of the column values.

```
SELECT SUM(AnnualRevenue) FROM [CData].[main].Customer WHERE CustomerId
= '12345'
```

Projection Functions

AVG([DISTINCT] expression)

The AVG function returns the average (arithmetic mean) of the input expression values. The AVG function works with numeric values and ignores NULL values.

- **expression:** The expression to use to compute the average.

COUNT([DISTINCT] expression)

The COUNT function counts the rows defined by the expression.

- **expression:** The expression to use to compute the count.

MAX([DISTINCT] expression)

The MIN function returns the minimum value in a set of rows. DISTINCT may be used but do not affect the result.

- **expression:** The expression to use to compute the max.

MIN([DISTINCT] expression)

The MIN function returns the minimum value in a set of rows. DISTINCT may be used but do not affect the result.

- **expression:** The expression to use to compute the min.

SUM([DISTINCT] expression)

Returns the sum on non-null values. Each distinct value of expression is aggregated only once into the result.

- **expression:** The expression to use to compute the sum.

COALESCE(expr1 [, expr2 [, ...]])

An NVL or COALESCE expression returns the value of the first expression in the list that is not null. If all expressions are null, the result is null. When a non-null value is found, the remaining expressions in the list are not evaluated.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

NVL(expr1 [, expr2 [, ...]])

An NVL or COALESCE expression returns the value of the first expression in the list that is not null. If all expressions are null, the result is null. When a non-null value is found, the remaining expressions in the list are not evaluated.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

DECODE(expression, search, result [, search2, result2], default)

A DECODE expression replaces a specific value with either another specific value or a default value, depending on the result of an equality condition. This operation is equivalent

to the operation of a simple CASE expression or an IF-THEN-ELSE statement.

- **expression:** The source of the value that you want to compare, such as a column in a table.
- **search:** The target value that is compared against the source expression, such as a numeric value or a character string. The search expression must evaluate to a single fixed value.
- **result:** The replacement value that query returns when the expression matches the search value.
- **search2:** The target value that is compared against the source expression, such as a numeric value or a character string. The search expression must evaluate to a single fixed value.
- **result2:** The replacement value that query returns when the expression matches the search value.
- **default:** An optional default value that is used for cases when the search condition fails. If you do not specify a default value, the DECODE expression returns NULL.

GREATEST(expr1 [, expr2 [, ...]])

Returns the largest value from a list of any number of expressions.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

LEAST(expr1 [, expr2 [, ...]])

Returns the smallest value from a list of any number of expressions.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

NULLIF(expression1, expression2)

The NULLIF expression compares two arguments and returns null if the arguments are equal. If they are not equal, the first argument is returned. This expression is the inverse of the NVL or COALESCE expression.

- **expression1:** The target columns or expressions that are compared.
- **expression2:** The target columns or expressions that are compared.

ADD_MONTHS(expression, num_months)

ADD_MONTHS adds the specified number of months to a date or timestamp value or expression.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.
- **num_months:** A positive or negative integer. Use a negative number to subtract months from dates.

CURRENT_DATE()

CURRENT_DATE returns a date in the current session time zone (UTC by default) in the default format: yyyy-MM-dd.

CURRENT_TIME()

CURRENT_TIME returns a time in the current session time zone (UTC by default) in the default format: hh:mm:ss.mmm.

CURRENT_TIMESTAMP()

CURRENT_TIMESTAMP returns a timestamp in the current session time zone (UTC by default) in the default format: yyyy-MM-dd'T'hh:mm:ss.mmm'Z'.

DATEADD(datepart, interval, expression)

Increments a date or timestamp value by a specified interval.

- **datepart:** The date part (year, month, or day, for example) that the function operates on.
- **interval:** An integer that specified the interval (number of days, for example) to add

to the target expression. A negative integer subtracts the interval.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp. The date or timestamp expression must contain the specified date part.

DATEDIFF(datepart, expression1, expression2)

DATEDIFF returns the difference between the date parts of two date or time expressions. If the second date or time is later than the first date or time, the result is positive. If the second date or time is earlier than the first date or time, the result is negative.

- **datepart:** The specific part of the date value (year, month, or day, for example) that the function operates on.
- **expression1:** A date or timestamp column or expression that implicitly converts to a date or timestamp. The expressions must both contain the specified date part.
- **expression2:** A date or timestamp column or expression that implicitly converts to a date or timestamp. The expressions must both contain the specified date part.

DATEPART(datepart, expression)

DATEPART extracts datepart values from an expression.

- **datepart:** The specific part of the date value (year, month, or day, for example) that the function operates on.
- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp. The expression must be a date or timestamp expression that contains the specified datepart.

TIMESTAMPADD(datetimepart, interval, expression)

Increments a date or timestamp value by a specified interval.

- **datetimepart:** The datetime part (year, quarter, month, week, day, hour, minute, or seconds for example) that the function operates on.
- **interval:** An integer that specified the interval (number of days, for example) to add to the target expression. A negative integer subtracts the interval.

- **expression:** A timestamp column or an expression that implicitly converts to a timestamp. The timestamp expression must contain the specified datetime part.

TIMESTAMPDIFF(datetimepart, expression1, expression2)

TIMESTAMPDIFF returns the difference between the date parts of two date or time expressions. If the second date or time is later than the first date or time, the result is positive. If the second date or time is earlier than the first date or time, the result is negative.

- **datepart:** The specific part of the date value (year, quarter, month, week, day, hour, minute, or seconds for example) that the function operates on.
- **expression1:** A timestamp column or expressions that implicitly converts to a timestamp. The expressions must both contain the specified datetime part.
- **expression2:** A timestamp column or expressions that implicitly converts to a timestamp. The expressions must both contain the specified datetime part.

DAY(date)

DAY returns the day of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYNAME(date)

DAYNAME returns the name of day of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYOFMONTH(date)

DAYOFMONTH returns the day of the month of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYOFWEEK(date)

DAYOFWEEK returns the day of the week of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYOFYEAR(date)

DAYOFYEAR returns the day of the year of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

LAST_DAY(expression)

LAST_DAY returns the date of the last day of the month that contains date.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

MONTHS_BETWEEN(date1, date2)

MONTHS_BETWEEN determines the number of months between two dates. If the first date is later than the second date, the result is positive; otherwise, the result is negative. If either argument is null, the result is NULL.

- **date1:** An expression, such as a column name, that evaluates to a valid date or timestamp value.
- **date2:** An expression, such as a column name, that evaluates to a valid date or timestamp value.

NEXT_DAY(expression, day)

NEXT_DAY returns the date of the first instance of the specified day that is later than the given date. If the day value is the same day of the week as given_date, the next occurrence of that day is returned.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.
- **day:** A string containing the name of any day. Capitalization does not matter.

HOUR(timestamp)

HOUR returns the hour of the specified timestamp.

- **timestamp:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

MINUTE(timestamp)

MINUTE returns the minute of the specified timestamp.

- **timestamp:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

SECOND(timestamp)

SECOND returns the seconds of the specified timestamp.

- **timestamp:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

WEEK(date)

WEEK returns the week of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

MONTH(date)

MONTH returns the month of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date

or timestamp.

MONTHNAME(date)

MONTHNAME returns the name of month of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

QUARTER(date)

QUARTER returns the quarter of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

YEAR(date)

YEAR returns the year of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

TRUNC(expression [, decimalplaces])

Truncates a timestamp and returns a date.

- **expression:** Can be a numeric data type or a timestamp column or an expression that implicitly converts to a timestamp.
- **decimalplaces:** An integer that indicates the number of decimal places of precision, in either direction.

ABS(expression)

ABS calculates the absolute value of a number, where that number can be a literal or an expression that evaluates to a number.

- **expression:** Number or expression that evaluates to a number.

ACOS(expression)

ACOS is a trigonometric function that returns the arc cosine of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

ASIN(expression)

ASIN is a trigonometric function that returns the arc sine of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

ATAN(expression)

ATAN is a trigonometric function that returns the arc tangent of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

ATAN2(expression1, expression2)

ATAN2 is a trigonometric function that returns the arc tangent of a one number divided by another number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression1:** A double precision number.
- **expression2:** A double precision number.

CEIL(expression)

The CEILING or CEIL function is used to round a number up to the next whole number.

- **expression:** A double precision number.

CEILING(expression)

The CEILING or CEIL function is used to round a number up to the next whole number.

- **expression:** A double precision number.

COS(expression)

COS is a trigonometric function that returns the cosine of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

COT(expression)

COT is a trigonometric function that returns the cotangent of a number. The input parameter must be nonzero.

- **expression:** A double precision number.

DEGREES(expression)

Converts an angle in radians to its equivalent in degrees.

- **expression:** A double precision number.

EXP(expression)

The EXP function returns the exponential value in scientific notation for a numeric expression.

- **expression:** The expression must be an INTEGER, DECIMAL, or DOUBLE PRECISION data type.

FLOOR(expression)

The FLOOR function rounds a number down to the next whole number.

- **expression:** A double precision number.

LN(expression)

Returns the natural logarithm of the input parameter.

- **expression:** The target column or expression that the function operates on.

MOD(number1, number2)

The MOD function returns a numeric result that is the remainder of two numeric parameters. The first parameter is divided by the second parameter.

- **number1:** The first input parameter is an INTEGER, SMALLINT, BIGINT, or DECIMAL number.
- **number2:** The second parameter is an INTEGER, SMALLINT, BIGINT, or DECIMAL number. The same data type rules apply to number2 as to number1.

PI()

The PI function returns the value of PI to 14 decimal places.

POW(expression1, expression2)

The POW function is an exponential function that raises a numeric expression to the power of a second numeric expression.

- **expression1:** Numeric expression to be raised. Must be an integer, decimal, or floating-point data type.
- **expression2:** Power to raise expression1. Must be an integer, decimal, or floating-point data type.

POWER(expression1, expression2)

The POWER function is an exponential function that raises a numeric expression to the power of a second numeric expression.

- **expression1:** Numeric expression to be raised. Must be an integer, decimal, or floating-point data type.
- **expression2:** Power to raise expression1. Must be an integer, decimal, or floating-point data type.

RADIANS(expression)

Converts an angle in degrees to its equivalent in radians.

- **expression:** A double precision number.

RANDOM()

The RANDOM function generates a random value between 0.0 and 1.0.

ROUND(expression [, decimalplaces])

The ROUND function rounds numbers to the nearest integer or decimal.

- **expression:** INTEGER, DECIMAL, and FLOAT data types are supported.
- **decimalplaces:** An integer to indicate the number of decimal places for rounding.

SIN(expression)

SIN is a trigonometric function that returns the sine of a number. The return value is between -1 and 1.

- **expression:** A double precision number.

SIGN(expression)

The SIGN function returns the sign (positive or negative) of a number. The result of the SIGN function is 1, -1, or 0 indicating the sign of the argument.

- **expression:** Number to be evaluated. The data type can be numeric or double precision.

SQRT(expression)

The SQRT function returns the square root of a numeric value.

- **expression:** The expression must have an integer, decimal, or floating-point data

type.

TAN(expression)

TAN is a trigonometric function that returns the tangent of a number. The input parameter must be a non-zero number (in radians).

- **expression:** A double precision number.

ASCII(expression)

ASCII returns the sequence of Unicode code points that constitute \$expression.

- **expression:** A string expression.

LEN(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

CHAR_LENGTH(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

CHARACTER_LENGTH(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

LENGTH(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

TEXTLEN(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

BIT_LENGTH(expression)

Returns the length of the specified string in bits.

- **expression:** A string expression.

CONCAT(str1, str2)

The CONCAT function concatenates two character string and returns the resulting string.

- **str1:** A string expression.
- **str2:** A string expression.

INITCAP(expression)

Capitalizes the first letter of each word in a specified string. INITCAP supports UTF-8 multibyte characters, up to a maximum of four bytes per character.

- **expression:** A string expression.

LEFT(expression, num)

Returns the specified number of leftmost characters from a character string.

- **expression:** A string expression.
- **num:** A positive integer.

RIGHT(expression, num)

Returns the specified number of rightmost characters from a character string.

- **expression:** A string expression.
- **num:** A positive integer.

LOWER(expression)

Converts a string to lowercase. LOWER supports UTF-8 multibyte characters, up to a maximum of four bytes per character.

- **expression:** A string expression.

RPAD(string1, length [, string2])

Appends characters to a string, based on a specified length.

- **string1:** A character string or an expression that evaluates to a character string, such as the name of a character column.
- **length:** An integer that defines the length of the result of the function.
- **string2:** One or more characters that are appended to string1. This argument is optional; if it is not specified, spaces are used.

LPAD(string1, length [, string2])

Prepends characters to a string, based on a specified length.

- **string1:** A character string or an expression that evaluates to a character string, such as the name of a character column.
- **length:** An integer that defines the length of the result of the function.
- **string2:** One or more characters that are prepended to string1. This argument is optional; if it is not specified, spaces are used.

LTRIM(expression)

The LTRIM function trims a leading spaces of a string.

- **expression:** The string column or expression to be trimmed.

TRIMLEADING(expression)

The TRIMLEADING function trims a leading spaces of a string.

- **expression:** The string column or expression to be trimmed.

OCTET_LENGTH(expression)

Returns the length of the specified string as the number of bytes.

- **expression:** A string expression.

POSITION(substring IN string)

Returns the location of the specified substring within a string.

- **substring:** The substring to search for within the string.
- **string:** The string or column to be searched.

REPEAT(expression, numrepeats)

Repeats a string the specified number of times. If the input parameter is numeric, REPEAT treats it as a string.

- **expression:** The first input parameter is the string to be repeated.
- **numrepeats:** An integer indicating the number of times to repeat the string.

REPLACE(expression, old_chars, new_chars)

Replaces all occurrences of a set of characters within an existing string with other specified characters.

- **expression:** CHAR or VARCHAR string to be searched search
- **old_chars:** CHAR or VARCHAR string to replace.
- **new_chars:** New CHAR or VARCHAR string replacing the old_string.

RTRIM(expression)

The RTRIM function trims trailing spaces of a string.

- **expression:** The string column or expression to be trimmed.

STRPOS(string, substring)

Returns the position of a substring within a specified string.

- **string:** The string to be searched.
- **substring:** The substring to search for within the string.

SUBSTRING(expression, start_position, number_chars)

Returns the characters extracted from a string based on the specified character position for a specified number of characters.

- **expression:** The string to be searched. Non-character data types are treated like a string.
- **start_position:** The position within the string to begin the extraction, starting at 1.
- **number_chars:** The number of characters to extract (the length of the substring).

TRANSLATE(expression, characters_to_replace, characters_to_substitute)

For a given expression, replaces all occurrences of specified characters with specified substitutes. Existing characters are mapped to replacement characters by their positions in the characters_to_replace and characters_to_substitute arguments.

- **expression:** The expression to be translated.
- **characters_to_replace:** A string containing the characters to be replaced.
- **characters_to_substitute:** A string containing the characters to substitute.

UPPER(expression)

Converts a string to uppercase. UPPER supports UTF-8 multibyte characters, up to a maximum of four bytes per character.

- **expression:** A string expression.

CAST(expression AS type)

Cast is used in a query to indicate that the result type of an expression should be converted to some other type.

- **expression:** The expression to cast.
- **type:** The type to cast the expression to.

USER()

Returns the user name of the current "effective" user of the database, as applicable to checking permissions. Usually, this user name will be the same as the session user; however, this can occasionally be changed by superusers.

Predicate Functions

AVG([DISTINCT] expression)

The AVG function returns the average (arithmetic mean) of the input expression values. The AVG function works with numeric values and ignores NULL values.

- **expression:** The expression to use to compute the average.

COUNT([DISTINCT] expression)

The COUNT function counts the rows defined by the expression.

- **expression:** The expression to use to compute the count.

MAX([DISTINCT] expression)

The MIN function returns the minimum value in a set of rows. DISTINCT may be used but do not affect the result.

- **expression:** The expression to use to compute the max.

MIN([DISTINCT] expression)

The MIN function returns the minimum value in a set of rows. DISTINCT may be used but do not affect the result.

- **expression:** The expression to use to compute the min.

SUM([DISTINCT] expression)

Returns the sum on non-null values. Each distinct value of expression is aggregated only once into the result.

- **expression:** The expression to use to compute the sum.

COALESCE(expr1 [, expr2 [, ...]])

An NVL or COALESCE expression returns the value of the first expression in the list that is not null. If all expressions are null, the result is null. When a non-null value is found, the remaining expressions in the list are not evaluated.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

NVL(expr1 [, expr2 [, ...]])

An NVL or COALESCE expression returns the value of the first expression in the list that is not null. If all expressions are null, the result is null. When a non-null value is found, the remaining expressions in the list are not evaluated.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

DECODE(expression, search, result [, search2, result2], default)

A DECODE expression replaces a specific value with either another specific value or a default value, depending on the result of an equality condition. This operation is equivalent

to the operation of a simple CASE expression or an IF-THEN-ELSE statement.

- **expression:** The source of the value that you want to compare, such as a column in a table.
- **search:** The target value that is compared against the source expression, such as a numeric value or a character string. The search expression must evaluate to a single fixed value.
- **result:** The replacement value that query returns when the expression matches the search value.
- **search2:** The target value that is compared against the source expression, such as a numeric value or a character string. The search expression must evaluate to a single fixed value.
- **result2:** The replacement value that query returns when the expression matches the search value.
- **default:** An optional default value that is used for cases when the search condition fails. If you do not specify a default value, the DECODE expression returns NULL.

GREATEST(expr1 [, expr2 [, ...]])

Returns the largest value from a list of any number of expressions.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

LEAST(expr1 [, expr2 [, ...]])

Returns the smallest value from a list of any number of expressions.

- **expr1:** Any valid expression.
- **expr2:** Any valid expression.

NULLIF(expression1, expression2)

The NULLIF expression compares two arguments and returns null if the arguments are equal. If they are not equal, the first argument is returned. This expression is the inverse of the NVL or COALESCE expression.

- **expression1:** The target columns or expressions that are compared.
- **expression2:** The target columns or expressions that are compared.

ADD_MONTHS(expression, num_months)

ADD_MONTHS adds the specified number of months to a date or timestamp value or expression.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.
- **num_months:** A positive or negative integer. Use a negative number to subtract months from dates.

CURRENT_DATE()

CURRENT_DATE returns a date in the current session time zone (UTC by default) in the default format: yyyy-MM-dd.

CURRENT_TIME()

CURRENT_TIME returns a time in the current session time zone (UTC by default) in the default format: hh:mm:ss.mmm.

CURRENT_TIMESTAMP()

CURRENT_TIMESTAMP returns a timestamp in the current session time zone (UTC by default) in the default format: yyyy-MM-dd'T'hh:mm:ss.mmm'Z'.

DATEADD(datepart, interval, expression)

Increments a date or timestamp value by a specified interval.

- **datepart:** The date part (year, month, or day, for example) that the function operates on.
- **interval:** An integer that specified the interval (number of days, for example) to add

to the target expression. A negative integer subtracts the interval.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp. The date or timestamp expression must contain the specified date part.

DATEDIFF(datepart, expression1, expression2)

DATEDIFF returns the difference between the date parts of two date or time expressions. If the second date or time is later than the first date or time, the result is positive. If the second date or time is earlier than the first date or time, the result is negative.

- **datepart:** The specific part of the date value (year, month, or day, for example) that the function operates on.
- **expression1:** A date or timestamp column or expression that implicitly converts to a date or timestamp. The expressions must both contain the specified date part.
- **expression2:** A date or timestamp column or expression that implicitly converts to a date or timestamp. The expressions must both contain the specified date part.

DATEPART(datepart, expression)

DATEPART extracts datepart values from an expression.

- **datepart:** The specific part of the date value (year, month, or day, for example) that the function operates on.
- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp. The expression must be a date or timestamp expression that contains the specified datepart.

TIMESTAMPADD(datetimepart, interval, expression)

Increments a date or timestamp value by a specified interval.

- **datetimepart:** The datetime part (year, quarter, month, week, day, hour, minute, or seconds for example) that the function operates on.
- **interval:** An integer that specified the interval (number of days, for example) to add to the target expression. A negative integer subtracts the interval.

- **expression:** A timestamp column or an expression that implicitly converts to a timestamp. The timestamp expression must contain the specified datetime part.

TIMESTAMPDIFF(datetimepart, expression1, expression2)

TIMESTAMPDIFF returns the difference between the date parts of two date or time expressions. If the second date or time is later than the first date or time, the result is positive. If the second date or time is earlier than the first date or time, the result is negative.

- **datepart:** The specific part of the date value (year, quarter, month, week, day, hour, minute, or seconds for example) that the function operates on.
- **expression1:** A timestamp column or expressions that implicitly converts to a timestamp. The expressions must both contain the specified datetime part.
- **expression2:** A timestamp column or expressions that implicitly converts to a timestamp. The expressions must both contain the specified datetime part.

DAY(date)

DAY returns the day of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYNAME(date)

DAYNAME returns the name of day of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYOFMONTH(date)

DAYOFMONTH returns the day of the month of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYOFWEEK(date)

DAYOFWEEK returns the day of the week of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

DAYOFYEAR(date)

DAYOFYEAR returns the day of the year of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

LAST_DAY(expression)

LAST_DAY returns the date of the last day of the month that contains date.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

MONTHS_BETWEEN(date1, date2)

MONTHS_BETWEEN determines the number of months between two dates. If the first date is later than the second date, the result is positive; otherwise, the result is negative. If either argument is null, the result is NULL.

- **date1:** An expression, such as a column name, that evaluates to a valid date or timestamp value.
- **date2:** An expression, such as a column name, that evaluates to a valid date or timestamp value.

NEXT_DAY(expression, day)

NEXT_DAY returns the date of the first instance of the specified day that is later than the given date. If the day value is the same day of the week as given_date, the next occurrence of that day is returned.

- **expression:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.
- **day:** A string containing the name of any day. Capitalization does not matter.

HOUR(timestamp)

HOUR returns the hour of the specified timestamp.

- **timestamp:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

MINUTE(timestamp)

MINUTE returns the minute of the specified timestamp.

- **timestamp:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

SECOND(timestamp)

SECOND returns the seconds of the specified timestamp.

- **timestamp:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

WEEK(date)

WEEK returns the week of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

MONTH(date)

MONTH returns the month of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date

or timestamp.

MONTHNAME(date)

MONTHNAME returns the name of month of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

QUARTER(date)

QUARTER returns the quarter of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

YEAR(date)

YEAR returns the year of the specified date.

- **date:** A date or timestamp column or an expression that implicitly converts to a date or timestamp.

TRUNC(expression [, decimalplaces])

Truncates a timestamp and returns a date.

- **expression:** Can be a numeric data type or a timestamp column or an expression that implicitly converts to a timestamp.
- **decimalplaces:** An integer that indicates the number of decimal places of precision, in either direction.

ABS(expression)

ABS calculates the absolute value of a number, where that number can be a literal or an expression that evaluates to a number.

- **expression:** Number or expression that evaluates to a number.

ACOS(expression)

ACOS is a trigonometric function that returns the arc cosine of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

ASIN(expression)

ASIN is a trigonometric function that returns the arc sine of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

ATAN(expression)

ATAN is a trigonometric function that returns the arc tangent of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

ATAN2(expression1, expression2)

ATAN2 is a trigonometric function that returns the arc tangent of a one number divided by another number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression1:** A double precision number.
- **expression2:** A double precision number.

CEIL(expression)

The CEILING or CEIL function is used to round a number up to the next whole number.

- **expression:** A double precision number.

CEILING(expression)

The CEILING or CEIL function is used to round a number up to the next whole number.

- **expression:** A double precision number.

COS(expression)

COS is a trigonometric function that returns the cosine of a number. The return value is in radians and is between $\pi/2$ and $-\pi/2$.

- **expression:** A double precision number.

COT(expression)

COT is a trigonometric function that returns the cotangent of a number. The input parameter must be nonzero.

- **expression:** A double precision number.

DEGREES(expression)

Converts an angle in radians to its equivalent in degrees.

- **expression:** A double precision number.

EXP(expression)

The EXP function returns the exponential value in scientific notation for a numeric expression.

- **expression:** The expression must be an INTEGER, DECIMAL, or DOUBLE PRECISION data type.

FLOOR(expression)

The FLOOR function rounds a number down to the next whole number.

- **expression:** A double precision number.

LN(expression)

Returns the natural logarithm of the input parameter.

- **expression:** The target column or expression that the function operates on.

MOD(number1, number2)

The MOD function returns a numeric result that is the remainder of two numeric parameters. The first parameter is divided by the second parameter.

- **number1:** The first input parameter is an INTEGER, SMALLINT, BIGINT, or DECIMAL number.
- **number2:** The second parameter is an INTEGER, SMALLINT, BIGINT, or DECIMAL number. The same data type rules apply to number2 as to number1.

PI()

The PI function returns the value of PI to 14 decimal places.

POW(expression1, expression2)

The POW function is an exponential function that raises a numeric expression to the power of a second numeric expression.

- **expression1:** Numeric expression to be raised. Must be an integer, decimal, or floating-point data type.
- **expression2:** Power to raise expression1. Must be an integer, decimal, or floating-point data type.

POWER(expression1, expression2)

The POWER function is an exponential function that raises a numeric expression to the power of a second numeric expression.

- **expression1:** Numeric expression to be raised. Must be an integer, decimal, or floating-point data type.
- **expression2:** Power to raise expression1. Must be an integer, decimal, or floating-point data type.

RADIANS(expression)

Converts an angle in degrees to its equivalent in radians.

- **expression:** A double precision number.

RANDOM()

The RANDOM function generates a random value between 0.0 and 1.0.

ROUND(expression [, decimalplaces])

The ROUND function rounds numbers to the nearest integer or decimal.

- **expression:** INTEGER, DECIMAL, and FLOAT data types are supported.
- **decimalplaces:** An integer to indicate the number of decimal places for rounding.

SIN(expression)

SIN is a trigonometric function that returns the sine of a number. The return value is between -1 and 1.

- **expression:** A double precision number.

SIGN(expression)

The SIGN function returns the sign (positive or negative) of a number. The result of the SIGN function is 1, -1, or 0 indicating the sign of the argument.

- **expression:** Number to be evaluated. The data type can be numeric or double precision.

SQRT(expression)

The SQRT function returns the square root of a numeric value.

- **expression:** The expression must have an integer, decimal, or floating-point data

type.

TAN(expression)

TAN is a trigonometric function that returns the tangent of a number. The input parameter must be a non-zero number (in radians).

- **expression:** A double precision number.

ASCII(expression)

ASCII returns the sequence of Unicode code points that constitute \$expression.

- **expression:** A string expression.

LEN(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

CHAR_LENGTH(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

CHARACTER_LENGTH(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

LENGTH(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

TEXTLEN(expression)

Returns the length of the specified string as the number of characters.

- **expression:** A string expression.

BIT_LENGTH(expression)

Returns the length of the specified string in bits.

- **expression:** A string expression.

CONCAT(str1, str2)

The CONCAT function concatenates two character string and returns the resulting string.

- **str1:** A string expression.
- **str2:** A string expression.

INITCAP(expression)

Capitalizes the first letter of each word in a specified string. INITCAP supports UTF-8 multibyte characters, up to a maximum of four bytes per character.

- **expression:** A string expression.

LEFT(expression, num)

Returns the specified number of leftmost characters from a character string.

- **expression:** A string expression.
- **num:** A positive integer.

RIGHT(expression, num)

Returns the specified number of rightmost characters from a character string.

- **expression:** A string expression.
- **num:** A positive integer.

LOWER(expression)

Converts a string to lowercase. LOWER supports UTF-8 multibyte characters, up to a maximum of four bytes per character.

- **expression:** A string expression.

RPAD(string1, length [, string2])

Appends characters to a string, based on a specified length.

- **string1:** A character string or an expression that evaluates to a character string, such as the name of a character column.
- **length:** An integer that defines the length of the result of the function.
- **string2:** One or more characters that are appended to string1. This argument is optional; if it is not specified, spaces are used.

LPAD(string1, length [, string2])

Prepends characters to a string, based on a specified length.

- **string1:** A character string or an expression that evaluates to a character string, such as the name of a character column.
- **length:** An integer that defines the length of the result of the function.
- **string2:** One or more characters that are prepended to string1. This argument is optional; if it is not specified, spaces are used.

LTRIM(expression)

The LTRIM function trims a leading spaces of a string.

- **expression:** The string column or expression to be trimmed.

TRIMLEADING(expression)

The TRIMLEADING function trims a leading spaces of a string.

- **expression:** The string column or expression to be trimmed.

OCTET_LENGTH(expression)

Returns the length of the specified string as the number of bytes.

- **expression:** A string expression.

POSITION(substring IN string)

Returns the location of the specified substring within a string.

- **substring:** The substring to search for within the string.
- **string:** The string or column to be searched.

REPEAT(expression, numrepeats)

Repeats a string the specified number of times. If the input parameter is numeric, REPEAT treats it as a string.

- **expression:** The first input parameter is the string to be repeated.
- **numrepeats:** An integer indicating the number of times to repeat the string.

REPLACE(expression, old_chars, new_chars)

Replaces all occurrences of a set of characters within an existing string with other specified characters.

- **expression:** CHAR or VARCHAR string to be searched search
- **old_chars:** CHAR or VARCHAR string to replace.
- **new_chars:** New CHAR or VARCHAR string replacing the old_string.

RTRIM(expression)

The RTRIM function trims trailing spaces of a string.

- **expression:** The string column or expression to be trimmed.

STRPOS(string, substring)

Returns the position of a substring within a specified string.

- **string:** The string to be searched.
- **substring:** The substring to search for within the string.

SUBSTRING(expression, start_position, number_chars)

Returns the characters extracted from a string based on the specified character position for a specified number of characters.

- **expression:** The string to be searched. Non-character data types are treated like a string.
- **start_position:** The position within the string to begin the extraction, starting at 1.
- **number_chars:** The number of characters to extract (the length of the substring).

TRANSLATE(expression, characters_to_replace, characters_to_substitute)

For a given expression, replaces all occurrences of specified characters with specified substitutes. Existing characters are mapped to replacement characters by their positions in the characters_to_replace and characters_to_substitute arguments.

- **expression:** The expression to be translated.
- **characters_to_replace:** A string containing the characters to be replaced.
- **characters_to_substitute:** A string containing the characters to substitute.

UPPER(expression)

Converts a string to uppercase. UPPER supports UTF-8 multibyte characters, up to a maximum of four bytes per character.

- **expression:** A string expression.

CAST(expression AS type)

Cast is used in a query to indicate that the result type of an expression should be converted to some other type.

- **expression:** The expression to cast.
- **type:** The type to cast the expression to.

USER()

Returns the user name of the current "effective" user of the database, as applicable to checking permissions. Usually, this user name will be the same as the session user; however, this can occasionally be changed by superusers.

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 Name, TotalDue INTO 'csv://c:/[CData].[main].Customer.txt' FROM '[CData].[main].Customer' WHERE CustomerId = '12345'");
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 '[CData].[main].Customer' IN 'csv://filename=c:/[CData].[main].Customer.csv;delimiter=tab' FROM '[CData].[main].Customer' WHERE CustomerId = '12345'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

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 Syntax

```
"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 adapter models template views created in the MarkLogic server as views. These definitions are dynamically retrieved. Any changes you make, such as adding a new table, or adding new columns, or changing the data type of a column, will immediately be reflected when you connect using the adapter.

Creating Template Views

1. Navigate to the query console, <http://Server:8000/qconsole>, where Server is the DSN of the Server MarkLogic is hosted.
2. Create a new query tab.
3. Set the Query Type to XQuery.
4. From there, follow the guide to create a template view from the official MarkLogic website here: <https://docs.marklogic.com/guide/sql/creating-template-views>.

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
User	The MarkLogic user account used to authenticate.
Password	The password used to authenticate the user.
Server	The address of the MarkLogic server to which you are connecting.
Port	The ODBC Server port for the MarkLogic.
Database	The name of the MarkLogic database to connect to.
API	Specifies the API which will be used by the provider to query data.
UseSSL	This field sets whether SSL is enabled.

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

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.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from MarkLogic.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.

Authentication

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

Property	Description
User	The MarkLogic user account used to authenticate.
Password	The password used to authenticate the user.
Server	The address of the MarkLogic server to which you are connecting.
Port	The ODBC Server port for the MarkLogic.
Database	The name of the MarkLogic database to connect to.
API	Specifies the API which will be used by the provider to query data.
UseSSL	This field sets whether SSL is enabled.

User

The MarkLogic user account used to authenticate.

Data Type

string

Default Value

""

Remarks

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

Server

The address of the MarkLogic server to which you are connecting.

Data Type

string

Default Value

""

Remarks

The address of the MarkLogic server to which you are connecting.

Port

The ODBC Server port for the MarkLogic.

Data Type

string

Default Value

""

Remarks

The ODBC Server port for the MarkLogic. If not specified the default port number 5432 is used for the ODBC server. If API=REST, then by default, port number 8000 will be used.

Database

The name of the MarkLogic database to connect to.

Data Type

string

Default Value

""

Remarks

The name of the MarkLogic database to connect to.

API

Specifies the API which will be used by the provider to query data.

Possible Values

ODBC, REST

Data Type

string

Default Value

"ODBC"

Remarks

Specifies the API which will be used by the provider to query data. Valid values are ODBC (to connect to the ODBC server) and REST (to connect to the REST API).

UseSSL

This field sets whether SSL is enabled.

Data Type

bool

Default Value

false

Remarks

This field sets whether the adapter will attempt to negotiate TLS/SSL connections to the server. By default, the adapter checks the server's certificate against the system's trusted certificate store. To specify another certificate, set [SSLServerCert](#).

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----- MIICHTCCAe4CAQAwDQYJKoZIhV.....Qw == -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	34e929226ae0819f2ec14b4a3d904f801c bb150d
The SHA1 Thumbprint (hex values can also be either space or colon separated)	34e929226ae0819f2ec14b4a3d904f801c 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 MarkLogic 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\\MarkLogic 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\\MarkLogic 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
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.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from MarkLogic.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.

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.

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.

Pagsize

The maximum number of results to return per page from MarkLogic.

Data Type

int

Default Value

1000

Remarks

The Pagsize property affects the maximum number of results to return per page from MarkLogic. Setting a higher value may result in better performance at the cost of additional memory allocated per page consumed.

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.

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 [CData].[main].Customer 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"
```

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.