



TIBCO® Data Virtualization

Splunk Adapter Guide

Version 8.7.0 | October 2023

Contents

Contents	2
Splunk Adapter	4
Getting Started	4
Basic Tab	5
Logging	5
Changelog	7
Advanced Features	11
User Defined Views	12
SSL Configuration	14
Firewall and Proxy	15
Query Processing	15
Logging	16
SQL Compliance	19
SELECT Statements	20
SELECT INTO Statements	25
INSERT Statements	26
UPDATE Statements	27
DELETE Statements	27
EXECUTE Statements	28
PIVOT and UNPIVOT	29
Data Model	30
Tables	31
Views	51
Stored Procedures	59
Connection String Options	61
Authentication	64
SSL	67

Firewall	69
Proxy	72
Logging	78
Schema	79
Miscellaneous	80
TIBCO Product Documentation and Support Services	89
How to Access TIBCO Documentation	89
How to Contact TIBCO Support	90
Release Version Support	90
How to Join TIBCO Community	91
Legal and Third-Party Notices	92

Splunk Adapter

Splunk Version Support

The adapter leverages the Splunk Rest API to enable you to access data models in Splunk Enterprise or Splunk Cloud as relational databases, enabling bidirectional access to reports, datasets, and table datasets.

Note: Splunk does not enable access to their API for users with free trial accounts. As a result, you must have a paid Splunk account to connect using the adapter.

SQL Compliance

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

Getting Started

Connecting to Splunk

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

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

1. Unzip the `tdv.splunk.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.splunk/tdv.splunk.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 Splunk
```

Basic Tab

Connecting to Splunk APIs

You must specify the URL to a valid Splunk server. By default the adapter makes requests on port 8089.

By default, the adapter attempts to negotiate TLS/SSL with the server. See [SSL Configuration](#) for more information on TLS/SSL configuration.

Authenticating to Splunk

Login with Splunk credentials is the only available authentication method for connecting to Splunk.

Splunk credentials

To authenticate with Splunk credentials, set the User and Password to your login credentials.

Splunk Token

To authenticate with Splunk token, set the AuthScheme to `AccessToken`; and AccessToken property to your token generated from Splunk UI under Users and Authentication/Tokens.

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 Splunk 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**.

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.
10/05/2022	8313	Splunk	Added <ul style="list-style-type: none"> Added support for authentication with Token. This Token is generated from Splunk UI under Users and Authentication/Tokens.
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

			<p>projection vs the predicate. I.e: <code>SELECT LAST_MONTH() AS lm, Col FROM Table WHERE Col > LAST_MONTH()</code>. Formerly the two <code>LAST_MONTH()</code> methods would resolve to different datetimes. Now they will match.</p> <ul style="list-style-type: none"> As a replacement for the previous behavior, the relative date/datetime functions in the criteria may have an 'L' appended to them. I.e: <code>WHERE col > L_LAST_MONTH()</code>. This will continue to resolve to the same values that previously were calculated in the criteria. Note that the "L_" prefix will only work in the predicate - it not available for the projection.
07/08/2021	7859	General	<p>Added</p> <ul style="list-style-type: none"> Added the TCP Logging Module for the logging information happening on the TCP wire protocol. The transport bytes that are incoming and ongoing will be logged at verbosity=5.
04/23/2021	7785	General	<p>Added</p> <ul style="list-style-type: none"> Added support for handling client side formulas during insert / update. For example: <code>UPDATE Table SET Col1 = Concat (Col1, " - ", Col2) WHERE Col2 LIKE 'A%'</code>
04/23/2021	7783	General	<p>Changed</p> <ul style="list-style-type: none"> Updated how display sizes are determined for varchar primary key and foreign key columns so they will match the reported length of the column.
04/16/2021	7776	General	<p>Added</p> <ul style="list-style-type: none"> Non-conditional updates between two columns is now available to all drivers. For

example: UPDATE Table SET Col1=Col2

Changed

- 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
-------------	------	---------	--

03/25/2021	7754	Splunk	Deprecated <ul style="list-style-type: none"> • AppEmbedLink, IdpSystemUserName, IdpSystemPassword, IdpUrl, OktaDomain, OktaAppEmbedLink, OktaApiToken, SSOProvider, SSOUser and SSOPassword are deprecated.
------------	------	--------	--

Replacements

- IdpUrl and OktaAppEmbedLink are replaced by SSOLoginURL.
-

- The OktaDomain no longer needs to be specified for any SSO connections as it can be extracted from SSOLoginURL.
 - SSOLoginURL replaces OktaAppEmbedLink and AppEmbedLink as it is more generic and applicable to other SSO providers.
 - SSouser, SSOPassword, IdpSystemUserName, IdpSystemPassword are replaced with the User / Password connection properties.
 - OktaApiToken is replaced by ApiToken.
-

Advanced Features

This section details a selection of advanced features of the Splunk 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 Splunk 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 Splunk 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 DataModels 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.

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

The Splunk 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 <u>Verbosity</u> 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 verbatimities, which can delay execution times.

To refine the logged content further by showing/hiding specific categories of information, see LogModules.

Sensitive Data

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

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

Best Practices for Data Security

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

Java Logging

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

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

Advanced Logging

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

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

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

- **EXEC:** Query Execution. Includes execution messages for original SQL queries, parsed SQL queries, and normalized SQL queries. Query and page success/failure messages appear here as well.
- **INFO:** General Information. Includes the connection string, driver version (build number), and initial connection messages.
- **HTTP:** HTTP Protocol messages. Includes HTTP requests/responses (including POST messages), as well as Kerberos related messages.
- **SSL :** SSL certificate messages.
- **OAUT:** OAuth related failure/success messages.
- **SQL :** Includes SQL transactions, SQL bulk transfer messages, and SQL result set messages.
- **META:** Metadata cache and schema messages.
- **TCP :** Incoming and Ongoing raw bytes on TCP transport layer messages.

An example value for this property would be.

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

Note that these modules refine the information as it is pulled after taking the [Verbosity](#) into account.

SQL Compliance

The Splunk Adapter supports several operations on data, including querying, deleting, modifying, and inserting.

SELECT Statements

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

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

INSERT Statements

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

UPDATE Statements

The primary key Id is required to update a record. See [UPDATE Statements](#) for a syntax reference and examples.

DELETE Statements

The primary key Id is required to delete a record. See [DELETE Statements](#) for a syntax reference and examples.

EXECUTE Statements

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

Names and Quoting

- Table and column names are considered identifier names; as such, they are restricted to the following characters: [A-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 Splunk 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>
  ]
} | SCOPE_IDENTITY()
<expression> ::=
  | <column_reference>
  | @ <parameter>
  | ?
  | COUNT( * | { [ DISTINCT ] <expression> } )
  | { AVG | MAX | MIN | SUM | COUNT } ( <expression> )
  | NULLIF ( <expression> , <expression> )
  | COALESCE ( <expression> , ... )
  | CASE <expression>

```

```

        WHEN { <expression> | <search_condition> } THEN { <expression> |
NULL } [ ... ]
    [ ELSE { <expression> | NULL } ]
    END
    | <literal>
    | <sql_function>
<search_condition> ::=
{
    <expression> { = | < | > | >= | <= | != | AND | OR | NOT | IN | IS
| NULL | IS | NOT | NULL } [ <expression> ]
    } [ { AND | OR } ... ]

```

Examples

1. Return all columns:

```
SELECT * FROM DataModels
```

2. Rename a column:

```
SELECT "Owner" AS MY_Owner FROM DataModels
```

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

```
SELECT CAST(DatasetLimiting AS VARCHAR) AS Str_DatasetLimiting FROM
DataModels
```

4. Search data:

```
SELECT * FROM DataModels WHERE Id = 'SampleDataset'
```

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

```
SELECT * FROM DataModels WHERE Id = 'SampleDataset';
```

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

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

7. Summarize data:

```
SELECT Owner, MAX(DatasetLimiting) FROM DataModels GROUP BY Owner
```

See [Aggregate Functions](#) for details.

8. Sort a result set in ascending order:

```
SELECT Name, Owner FROM DataModels ORDER BY Owner ASC
```

Projection Functions

AVG([DISTINCT] expression)

Returns the average of the values of field expression.

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

COUNT([DISTINCT] expression)

Returns the number of occurrences of the field expression. To indicate a specific field value to match, format expression as `eval(field="value")`.

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

EARLIEST(expression)

Returns the chronologically earliest seen value of expression.

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

LATEST(expression)

Returns the chronologically latest seen value of expression.

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

MAX([DISTINCT] expression)

Returns the maximum value of the field expression. If the values of expression are non-numeric, the max is found from alphabetical ordering.

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

MEDIAN(expression)

Returns the middle-most value of the field.

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

MIN([DISTINCT] expression)

Returns the minimum value of the field expression. If the values of expression are non-numeric, the min is found from alphabetical ordering.

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

MODE(expression)

Returns the most frequent value of the field expression.

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

RANGE(expression)

Returns the difference between the max and min values of the field expression.

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

SUM([DISTINCT] expression)

Returns the sum of the values of the field expression.

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

SUMSQ(expression)

Returns the sum of the squares of the values of the field expression.

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

STDEV(expression)

Returns the sample standard deviation of the field expression.

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

STDEVP(expression)

Returns the population standard deviation of the field expression.

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

VAR(expression)

Returns the sample variance of the field expression.

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

VARP(expression)

Returns the population variance of the field expression.

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

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, Owner INTO
'csv://c:/DataModels.txt' FROM 'DataModels' WHERE Id =
'SampleDataset'");
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 'DataModels' IN
'csv://filename=c:/DataModels.csv;delimiter=tab' FROM 'DataModels' WHERE
Id = 'SampleDataset'");
System.out.println(stat.getUpdateCount()+" rows affected");
```

INSERT Statements

To create new records, use INSERT statements.

INSERT Syntax

The INSERT statement specifies the columns to be inserted and the new column values. You can specify the column values in a comma-separated list in the VALUES clause, as shown in the following example:

```
INSERT INTO <table_name>
( <column_reference> [ , ... ] )
VALUES
( { <expression> | NULL } [ , ... ] )

<expression> ::=
| @ <parameter>
| ?
| <literal>
```

You can use the executeUpdate method of the Statement and PreparedStatement classes to execute data manipulation commands and retrieve the rows affected.

```
String cmd = "INSERT INTO DataModels (Owner) VALUES (?);";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "user");
int count = pstmt.executeUpdate();
System.out.println(count+" rows were affected");
connection.close();
```

UPDATE Statements

To modify existing records, use UPDATE statements.

Update Syntax

The UPDATE statement takes as input a comma-separated list of columns and new column values as name-value pairs in the SET clause, as shown in the following example:

```
UPDATE <table_name> SET { <column_reference> = <expression> } [ , ... ]
WHERE { Id = <expression> } [ { AND | OR } ... ]
<expression> ::=
    | @ <parameter>
    | ?
    | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the rows affected, as shown in the following example:

```
String cmd = "UPDATE DataModels SET Owner='user' WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count = pstmt.executeUpdate();
System.out.println(count + " rows were affected");
connection.close();
```

DELETE Statements

To delete information from a table, use DELETE statements.

DELETE Syntax

The DELETE statement requires the table name in the FROM clause and the row's primary key in the WHERE clause, as shown in the following example:

```
<delete_statement> ::= DELETE FROM <table_name> WHERE { Id =
<expression> } [ { AND | OR } ... ]
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

You can use the executeUpdate method of the Statement or PreparedStatement classes to execute data manipulation commands and retrieve the number of affected rows, as shown in the following example:

```
Connection connection = DriverManager.getConnection
("jdbc:splunk:user=MyUserName;password=MyPassword;URL=MyURL;");
String cmd = "DELETE FROM DataModels WHERE Id = ?";
PreparedStatement pstmt = connection.prepareStatement(cmd);
pstmt.setString(1, "1");
int count=pstmt.executeUpdate();
connection.close();
```

EXECUTE Statements

To execute stored procedures, you can use EXECUTE or EXEC statements.

EXEC and EXECUTE assign stored procedure inputs, referenced by name, to values or parameter names.

Stored Procedure Syntax

To execute a stored procedure as an SQL statement, use the following syntax:

```
{ EXECUTE | EXEC } <stored_proc_name>
{
  [ @ ] <input_name> = <expression>
} [ , ... ]
<expression> ::=
  | @ <parameter>
  | ?
  | <literal>
```

Example Statements

Reference stored procedure inputs by name:

```
EXECUTE my_proc @second = 2, @first = 1, @third = 3;
```

Execute a parameterized stored procedure statement:

```
EXECUTE my_proc second = @p1, first = @p2, third = @p3;
```

PIVOT and UNPIVOT

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

PIVOT

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

PIVOT Syntax

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

UNPIVOT

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

UNPIVOT Sytax

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

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

Data Model

The adapter models Splunk reports, searches, datasets, and data models as tables in a relational database that you can read from/write to with SQL-92 queries.

Dynamic Schema Generation

You can work with all of the tables in your account: when you connect the adapter retrieves the metadata from Splunk and dynamically reflects any changes in the table schemas.

You can call the [CreateSchema](#) stored procedure to persist a static schema across connections. The stored procedure saves the schema to a text file; the text file has a simple format that also makes schemas easy to customize.

Tables

See [Tables](#) for more details on updating and querying datasets, data models, and searches.

Views

The adapter also surfaces data through [Views](#) representing the following Splunk objects:

- **Reports:** See [LookUpReport](#) for an example of a view representing a saved report in

Splunk.

- **Data-model datasets:** See [AlertsInInternalServer](#) for an example of a view representing a dataset. See [Datasets](#) to retrieve a list of dataset views.
- **Table-type datasets:** See [UploadedModel](#) for an example of a view representing a table dataset in Splunk.

Tables

The adapter models the data in Splunk into a list of tables that can be queried using standard SQL statements.

Generally, querying Splunk tables is the same as querying a table in a relational database. Sometimes there are special cases, for example, including a certain column in the WHERE clause might be required to get data for certain columns in the table. This is typically needed for situations where a separate request must be made for each row to get certain columns. These types of situations are clearly documented at the top of the table page linked below.

Splunk Adapter Tables

Name	Description
DataModels	Create, query, update, and delete data models in Splunk.
Datasets	Create, query, update, and delete datasets in Splunk.
SearchJobs	Create, query, update, and delete search jobs in Splunk.

DataModels

Create, query, update, and delete data models in Splunk.

Select

The adapter will use the Splunk API to process search criteria that refer to the Id column. This column supports server-side processing for the = operator. The adapter processes other filters client-side within the adapter.

For example, the following query is processed server side by the Splunk APIs:

```
SELECT * FROM DataModels WHERE Id = 'SampleModel'
```

Insert

The Id column is the minimum requirement for an insert. In an insert, the DataModels table allows only the Id and Acceleration columns.

```
INSERT INTO DataModels (Id, Acceleration) VALUES ('initialname', '
{"enabled":false,"earliest_time":"","hunk.file_format":"","hunk.dfs_
block_size":0,"hunk.compression_codec":""}' )
```

Update

The DataModels table allows updates for the Acceleration column when Id is specified. You can also set the Provisional pseudocolumn.

```
UPDATE DataModels SET Provisional = 'true', Acceleration = '
{"enabled":false,"earliest_time": "-1mon", "cron_schedule": "0 */12 * *
*","hunk.file_format":"","hunk.dfs_block_size":0,"hunk.compression_
codec":""}' WHERE Id = 'initialname'
```

Delete

The DataModels table allows deleting a record when Id is specified.

```
DELETE FROM DataModels WHERE Id = 'initialname'
```

Columns

Name	Type	ReadOnly	References	Description
Id [KEY]	<i>String</i>	False		Id of the data model.
LinkId	<i>String</i>	True		Link of the data model.
Disabled	<i>Boolean</i>	True		Indicates if the data model is disabled/enabled.
UpdatedAt	<i>Datetime</i>	True		Datetime of the last update of the data model.
Description	<i>String</i>	True		Description of the data model.
Name	<i>String</i>	True		The name displayed for the data model in Splunk.
Author	<i>String</i>	True		Splunk user who created the data model.
App	<i>String</i>	True		Splunk app where the data model is shared.
Owner	<i>String</i>	True		Splunk user who owns the data model.
CanShareApp	<i>Boolean</i>	True		Boolean indicating whether the data model can be

			shared in an app.
CanShareGlobal	<i>Boolean</i>	True	Boolean indicating whether the data model can be shared globally.
CanShareUser	<i>Boolean</i>	True	Boolean indicating whether the data model can be shared by the user.
CanWrite	<i>Boolean</i>	True	Boolean indicating whether the data model can be extended by the user.
Modifiable	<i>Boolean</i>	True	Boolean indicating whether the data model can be modified.
Removable	<i>Boolean</i>	True	Boolean indicating whether the data model can be removed.
Acceleration	<i>String</i>	False	Acceleration settings for the data model. Supply JSON to specify any or all of the following settings: enabled (true or false), earliest_time (time modifier), or cron_schedule

			(cron string).
AccelerationAllowed	<i>Boolean</i>	True	Boolean indicating that acceleration is allowed or not for the data model.
AccelerationHunkCompression	<i>String</i>	True	Specifies the compression codec to be used for the accelerated orc or parquet format files.
DatasetCommands	<i>String</i>	True	Data model commands.
DatasetDescription	<i>String</i>	True	The JSON describing the data model.
DatasetCurrentCommand	<i>Integer</i>	True	Current command of the data model.
DatasetEarliestTime	<i>Datetime</i>	True	Earliest time of data model events being processed.
DatasetLatestTime	<i>Datetime</i>	True	Latest time of data model events being processed.
DatasetDiversity	<i>String</i>	True	Diversity of events being processed.
DatasetLimiting	<i>Integer</i>	True	Limitations of events being processed.

DatasetMode	<i>String</i>	True	Search mode events being processed.
DatasetSampleRatio	<i>String</i>	True	Sample ratio of the data model.
DatasetFields	<i>String</i>	True	Indexed fields the data model has.
DatasetType	<i>String</i>	True	Dataset type.
Type	<i>String</i>	True	Data model type.
Digest	<i>String</i>	True	Content digest type.
TagsWhitelist	<i>String</i>	True	Whitelist of data model tags.
ReadPermissions	<i>String</i>	True	Permissions to read this data model.
WritePermissions	<i>String</i>	True	Permissions to write to this data model.
Sharing	<i>String</i>	True	Data model sharing type.
Username	<i>String</i>	True	Username of the Splunk user.

Pseudo-Columns

Pseudo column fields are used in the WHERE clause of SELECT statements and offer a more granular control over the tuples that are returned from the data source.

Name	Type	Description
Provisional	<i>Boolean</i>	Indicates whether the data model is provisional. Provisional data models are not saved. Specify true to validate a data model before saving it.

Datasets

Create, query, update, and delete datasets in Splunk.

Select

The Datasets table requires DataModelId in the WHERE clause. The DataModelId column supports server-side processing for the = operator. The adapter processes other search criteria client-side within the adapter.

```
SELECT * FROM DataSets WHERE DataModelId = 'SampleModel'
```

Insert

Splunk allows inserts only when DataModelId, ParentName, and ObjectName are all specified.

```
INSERT INTO [Datasets] (ObjectName, ParentName, DataModelId) VALUES  
( 'SampleSet', 'BaseEvent', 'SampleModel' )
```

Update

The Datasets table allows updates when DataModelId is specified. The columns that can be updated in this case are the following: Description and DisplayName.

When ObjectName is also specified, you can update the following columns:
ObjectDisplayName, ParentName, Comment, Fields, Calculations, Constraints, Lineage,

ObjectSearchNoFields, ObjectSearch, AutoextractSearch, PreviewSearch, AccelerationSearch, BaseSearch, and TsidxNamespace.

```
UPDATE Datasets SET Description = 'model description' , DisplayName =
'Model Display Name' WHERE DataModelId = 'SampleModel'
UPDATE Datasets SET ParentName = 'BaseEvent', BaseSearch = '| search
(index=* OR index=*) | fields _time, RootObject', AccelerationSearch =
' search (index=* OR index=*) ' WHERE DataModelId = 'SampleModel' AND
ObjectName = 'SampleSet'
```

Delete

Datasets can be deleted by providing the DataModelId and the ObjectName of the dataset.

```
DELETE FROM Datasets WHERE DataModelId = 'SampleModel' AND ObjectName =
'SampleSet'
```

Columns

Name	Type	ReadOnly	References	Description
ObjectName [KEY]	String	False		Name of the dataset object.
DatamodelId [KEY]	String	False	DataModels.Id	Id of the data model the object belongs to.
DisplayName	String	False		Name of the data model the object belongs to.
Description	String	False		Dataset description.
ObjectNameList	String	True		List of the objects in the data model.

ObjectDisplayName	<i>String</i>	False	Name displayed in Splunk for the object.
ParentName	<i>String</i>	False	Name of the Parent Event.
Comment	<i>String</i>	False	Dataset comments.
Fields	<i>String</i>	False	Dataset events indexed fields.
Calculations	<i>String</i>	False	Saved calculations for dataset fields.
Constraints	<i>String</i>	False	Saved constraints for dataset fields.
Lineage	<i>String</i>	False	Dataset lineage.
ObjectSearchNoFields	<i>String</i>	False	Object search query without fields.
ObjectSearch	<i>String</i>	False	Saved search query for the object.
AutoextractSearch	<i>String</i>	False	Search query for autoextraction.
PreviewSearch	<i>String</i>	False	Search preview query.
AccelerationSearch	<i>String</i>	False	Search query including acceleration.
BaseSearch	<i>String</i>	False	Basic search query.

TsidxNamespace	<i>String</i>	False	Allocated namespace.
EventBased	<i>Integer</i>	True	Number of Event-Based objects in the data model.
TransactionBased	<i>Integer</i>	True	Number of Transaction-Based objects in the data model.
SearchBased	<i>Integer</i>	True	Number of Search-Based objects in the data model.

SearchJobs

Create, query, update, and delete search jobs in Splunk.

Select

The adapter will use the Splunk APIs to process the search Id (Sid) criteria specified in the WHERE clause. The Sid column supports server-side processing for the = operator. The adapter processes other search criteria client-side within the adapter.

```
SELECT * FROM SearchJobs
SELECT * FROM SearchJobs WHERE Sid = '123456789.1234'
```

Insert

Splunk allows inserts only when EventSearch is specified. You can insert the Custom, EarliestTime, LatestTime, Label, and StatusBuckets columns and all pseudocolumns.

```
INSERT Into SearchJobs (Custom, EventSearch, LatestTime, Timeout) VALUES
('custom1=test1, custom2=test2', ' from datamodel SampleModel', 'now',
'60')
```


Update

The SearchJobs table allows updates of the Custom column only when Sid is specified.

```
UPDATE SearchJobs SET Custom = 'custom1=test3, custom2=test4' WHERE sid = '123456789.1234'
```

Delete

SearchJobs can be deleted by providing the Sid.

```
DELETE FROM SearchJobs WHERE Sid = '123456789.1234'
```

Columns

Name	Type	ReadOnly	References	Description
Sid [KEY]	String	False		The search Id number.
EventSearch	String	False		Subset of the entire search that is before any transforming commands.
Custom	String	False		Custom job property. In an INSERT operation, pass the values as a comma-separated list of pairs of keys and values.

EarliestTime	<i>String</i>	False	The earliest time a search job is configured to start.
LatestTime	<i>String</i>	False	The latest time a search job is configured to start.
CursorTime	<i>String</i>	True	The earliest time from which no events are later scanned. Can be used to indicate progress.
Delegate	<i>String</i>	True	For saved searches, specifies jobs that were started by the user. Defaults to scheduler.
DiskUsage	<i>Long</i>	True	The total amount of disk space used, in bytes.
DispatchState	<i>String</i>	True	The state of the search. Can be any of QUEUED, PARSING, RUNNING, PAUSED, FINALIZING, FAILED, or DONE.
DoneProgress	<i>Double</i>	True	A number between 0 and

			1.0 that indicates the approximate progress of the search. doneProgress = (latestTime-cursorTime) / (latestTime-earliestTime)
DropCount	<i>Integer</i>	True	For real-time searches only, the number of possible events that were dropped due to the rt_queue_size (defaults to 100000).
EventAvailableCount	<i>Integer</i>	True	The number of events that are available for export.
EventCount	<i>Integer</i>	True	The number of events returned by the search.
EventFieldCount	<i>Integer</i>	True	The number of fields found in the search results.
EventIsStreaming	<i>Boolean</i>	True	Indicates if the events of this search are being streamed.
EventIsTruncated	<i>Boolean</i>	True	Indicates if the

			events of the search are not stored, making them unavailable from the events endpoint for the search.
EventPreviewableCount	<i>Integer</i>	True	Number of in-memory events that are not yet committed to disk.
EventSorting	<i>String</i>	True	Indicates if the events of this search are sorted, and in which order.
IsDone	<i>Boolean</i>	True	Indicates if the search has completed.
IsEventsPreviewEnabled	<i>String</i>	True	Indicates if the timeline_events_preview setting is enabled in limits.conf.
IsFailed	<i>Boolean</i>	True	Indicates if there was a fatal error executing the search. For example, invalid search string syntax.
IsFinalized	<i>Boolean</i>	True	Indicates if the search was

			finalized (stopped before completion).
IsPaused	<i>Boolean</i>	True	Indicates if the search is paused.
IsPreviewEnabled	<i>Boolean</i>	True	Indicates if previews are enabled.
IsRealTimeSearch	<i>Boolean</i>	True	Indicates if the search is a real-time search.
IsRemoteTimeline	<i>Boolean</i>	True	Indicates if the remote timeline feature is enabled.
IsSaved	<i>Boolean</i>	True	Indicates that the search job is saved on disk. Search artifacts are saved on disk for 7 days from the last time that the job was viewed or touched.
IsSavedSearch	<i>Boolean</i>	True	Indicates if this is a saved search run using the scheduler.
IsZombie	<i>Boolean</i>	True	Indicates if the process running the search died without finishing

			the search.
Keywords	<i>String</i>	True	All positive keywords used by this search. A positive keyword is a keyword that is not in a NOT clause.
Label	<i>String</i>	False	Custom name created for this search.
Messages	<i>String</i>	True	Errors and debug messages.
NumPreviews	<i>Integer</i>	True	Number of previews generated so far for this search job.
Performance	<i>String</i>	True	A representation of the execution costs.
Priority	<i>Integer</i>	True	An integer between 0-10 that indicates the search priority.
RemoteSearch	<i>String</i>	True	The search string that is sent to every search peer.
ReportSearch	<i>String</i>	True	If reporting commands are used, the

			reporting search.
ResultCount	<i>Integer</i>	True	The total number of results returned by the search. In other words, this is the subset of scanned events (represented by the ScanCount) that actually matches the search terms.
ResultsStreaming	<i>Boolean</i>	True	Indicates if the final results of the search are available using streaming (for example, no transforming operations).
ResultPreviewCount	<i>Integer</i>	True	The number of result rows in the latest preview results.
RunDuration	<i>Decimal</i>	True	Time in seconds that the search took to complete.
ScanCount	<i>Integer</i>	True	The number of events that are scanned or read off disk.
SearchEarliestTime	<i>Datetime</i>	True	Specifies the earliest time for a

			search, as specified in the search command rather than the EarliestTime parameter. It does not snap to the indexed data time bounds for all-time searches.
SearchLatestTime	<i>Datetime</i>	True	Specifies the latest time for a search, as specified in the search command rather than the LatestTime parameter. It does not snap to the indexed data time bounds for all-time searches.
SearchProviders	<i>String</i>	True	A list of all the search peers that were contacted.
StatusBuckets	<i>Integer</i>	False	Maximum number of timeline buckets.
TTL	<i>String</i>	True	The time to live, or the time before the search job expires after it completes.

Pseudo-Columns

Pseudo column fields are used in the WHERE clause of SELECT statements and offer a more granular control over the tuples that are returned from the data source.

Name	Type	Description
SearchMode	<i>String</i>	Searching mode, realtime or normal. If set to realtime, the search runs over the live data. The allowed values are <i>normal</i> , <i>realtime</i> .
EnableLookups	<i>Boolean</i>	Indicates whether lookups should be applied to events.
AutoPause	<i>Integer</i>	If specified, the search job pauses after this many seconds of inactivity. (0 means never autopause.)
AutoCancel	<i>Integer</i>	If specified, the job automatically cancels after this many seconds of inactivity. (0 means never autocancel.)
AdhocSearchLevel	<i>Integer</i>	Specify a search mode. Use one of the following search modes: verbose, fast, or smart. The allowed values are <i>verbose</i> , <i>fast</i> , <i>smart</i> .
ForceBundleReplication	<i>Boolean</i>	Specifies whether this search should cause (and wait depending on the value of SyncBundleReplication) for bundle synchronization with all search peers.
IndexEarliest	<i>String</i>	Specify a time string. Sets the earliest inclusive time bounds for the search, based on the index time bounds.
IndexLatest	<i>String</i>	Specify a time string. Sets the latest exclusive time bounds for the search, based on the index time bounds.
IndexedRealtime	<i>Boolean</i>	Indicates whether or not to use the indexed-realtime mode for real-time searches.

IndexedRealtimeOffset	<i>Integer</i>	Sets disk sync delay for indexed real-time search (seconds).
MaxCount	<i>Integer</i>	The number of events that can be accessible in any given status bucket.
MaxTime	<i>Integer</i>	Comma-separated list of (possibly wildcarded) servers from which raw events should be pulled.
Namespace	<i>String</i>	The application namespace in which to restrict searches.
Now	<i>String</i>	Specify a time string to set the absolute time used for any relative time specifier in the search. Defaults to the current system time. You can specify a relative time modifier for this parameter. For example, specify +2d to specify the current time plus two days.
ReduceFrequency	<i>Integer</i>	Determines how frequently to run the MapReduce reduce phase on accumulated map values.
ReloadMacros	<i>Boolean</i>	Specifies whether to reload macro definitions from the configuration file.
RemoteServerList	<i>Integer</i>	The number of seconds to run this search before finalizing. Specify 0 to never finalize.
ReplaySpeed	<i>Integer</i>	Indicate a real-time search replay speed factor. For example, 1 indicates normal speed, 0.5 indicates half of normal speed, and 2 indicates twice as fast as normal.
ReplayStartTime	<i>String</i>	Relative wall-clock start time for the replay.
ReplayEndTime	<i>String</i>	Relative end time for the replay clock. The replay stops when the clock time reaches this time.
ReuseMaxSecondsAgo	<i>Integer</i>	Specifies the number of seconds ago to check when

		an identical search is started and return the search Id of the job instead of starting a new job.
RequiredField	<i>String</i>	Adds a required field to the search.
RealTimeBlocking	<i>Boolean</i>	For a real-time search, indicates if the indexer blocks if the queue for this search is full.
RealTimeIndexFilter	<i>Boolean</i>	For a real-time search, indicates if the indexer prefilters events.
RealTimeMaxBlockSecs	<i>Integer</i>	For a real-time search with RealTimeBlocking set to true, the maximum time to block. Specify 0 to indicate no limit.
RealTimeQueueSize	<i>Integer</i>	For a real-time search, the queue size (in events) that the indexer should use for this search.
Timeout	<i>Integer</i>	The number of seconds to keep this search after processing has stopped.
SyncBundleReplication	<i>String</i>	Specifies whether this search should wait for bundle replication to complete.

Views

Views are composed of columns and pseudo columns. Views are similar to tables in the way that data is represented; however, views do not support updates. Entities that are represented as views are typically read-only entities. Often, a stored procedure is available to update the data if such functionality is applicable to the data source.

Queries can be executed against a view as if it were a normal table, and the data that comes back is similar in that regard.

Dynamic views, such as queries exposed as views, and views for looking up specific combinations of project_team work items are supported.

Splunk Adapter Views

Name	Description
AlertsInInternalServer	A dataset object in the example InternalServer data model.
LookUpReport	An example lookup report representing a view based on a saved report in Splunk.
UploadedModel	An example of a table object inside a data model.

AlertsInInternalServer

A dataset object in the example InternalServer data model.

Select

This is an example of a dataset view. These views are generated from dataset objects inside a data model. The adapter will use the Splunk APIs to process the following query components; the adapter processes other parts of the query client-side in memory.

All columns support server-side processing for the following operators and functions:

- Operators: =, <, >, >=, <=, IN, IS NULL, IS NOT NULL, NOT
- Functions: AVG, SUM, MIN, MAX, COUNT, STDEV, STDEVP, VAR, VARP

LIMIT, ORDER BY, GROUP BY, and HAVING are also processed server-side. An exception is the case when in the selected columns, there are fields that are not in the GROUP BY, and GROUP BY, criteria, and limiting are handled client-side.

In the case when an unsupported criteria or function is used, all processing will be completed client-side (except selecting specified fields). This is also the case when a SELECT statement has a column that is not in the GroupBy clause.

For example, the adapter uses the Splunk APIs to process the following queries.

```
SELECT Component, Timeendpos as Timeend FROM [AlertsInInternalServer]
WHERE Component = 'Saved' OR EventType != '' AND Priority IS NOT NULL
AND Linecount NOT IN ('1', '2') ORDER BY Priority DESC LIMIT 5
SELECT AVG(Suppressed), Priority FROM [AlertsInInternalServer] GROUP BY
Priority HAVING AVG(Suppressed) > 0
```

Columns

Name	Type	Description
_time	<i>Datetime</i>	
component	<i>String</i>	
date_hour	<i>Int</i>	
date_mday	<i>Int</i>	
date_minute	<i>Int</i>	
date_month	<i>String</i>	
date_second	<i>Int</i>	
date_wday	<i>String</i>	
date_year	<i>Int</i>	
date_zone	<i>Int</i>	
digest_mode	<i>Int</i>	
dispatch_time	<i>Int</i>	

host	<i>String</i>
linecount	<i>Int</i>
log_level	<i>String</i>
priority	<i>String</i>
punct	<i>String</i>
savedsearch_id	<i>String</i>
scheduled_time	<i>Int</i>
search_type	<i>String</i>
server_alert_actions	<i>String</i>
server_app	<i>String</i>
server_message	<i>String</i>
server_result_count	<i>Int</i>
server_run_time	<i>Double</i>
server_savedsearch_name	<i>String</i>

server_sid	String
server_status	String
server_user	String
source	String
sourcetype	String
splunk_server	String
suppressed	Int
thread_id	String
timeendpos	Int
timestartpos	Int
window_time	Int

LookUpReport

An example lookup report representing a view based on a saved report in Splunk.

Select

This is an example of a report view. These views are generated from saved reports in Splunk.

The adapter will use the Splunk APIs to process the following query components; the adapter processes other parts of the query client-side in memory.

Saved Search

Runs a saved search, or report, and returns the search results of a saved search. If the search contains replacement placeholder terms, such as \$replace_me\$, the search processor replaces the placeholders with the strings you specify.

For example:

- `SELECT * FROM mysearch WHERE replace_me='value'`

Will generate the following search statement:

- `| savedsearch mysearch replace_me="value"`

All replacement placeholder terms will be dynamic and saved as Pseudo-Columns.

All columns support server-side processing for the following operators and functions:

- Operators: =, <, >, >=, <=, IN, IS NULL, IS NOT NULL, NOT, LIKE
- Functions: AVG, SUM, MIN, MAX, COUNT, STDEV, STDEVP, VAR, VARP

LIMIT, ORDER BY, GROUP BY, and HAVING are also processed server-side. An exception is the case when in the selected columns, there are fields that are not in the GROUP BY, and GROUP BY, criteria, and limiting are handled client-side.

In the case when an unsupported criteria or function is used, all processing will be completed client-side (except selecting specified fields). This is also the case when a SELECT statement has a column that is not in the GROUP BY clause.

For example, the adapter processes the following queries server-side:

```
SELECT Country, Subregion as Sub FROM LookUpReport WHERE Iso2 != '123'
OR continent = 'Europe' AND iso3 NOT IN ('example_1', 'example_2') ORDER
BY Country DESC LIMIT 5
SELECT AVG(Iso2), Subregion FROM LookUpReport GROUP BY Subregion HAVING
AVG(Iso2) > 0
```


Columns

Name	Type	Description
continent	<i>String</i>	
country	<i>String</i>	
iso2	<i>String</i>	
iso3	<i>String</i>	
region_un	<i>String</i>	
region_wb	<i>String</i>	
subregion	<i>String</i>	

UploadedModel

An example of a table object inside a data model.

Select

This is an example of a view generated from a table object inside a data model. The adapter will use the Splunk APIs to process the following query components; the adapter processes other parts of the query client-side in memory.

All columns support server-side processing for the following operators and functions.

- Operators: =, <, >, >=, <=, IN, IS NULL, IS NOT NULL, NOT

- Functions: AVG, SUM, MIN, MAX, COUNT, STDEV, STDEVP, VAR, VARP

LIMIT, ORDER BY, GROUP BY, and HAVING are also processed server-side. An exception is the case when in the selected columns, there are fields that are not in the GROUP BY, and GROUP BY, criteria, and limiting are handled client-side.

In the case when an unsupported criteria or function is used, all processing will be completed client-side (except selecting specified fields). This is also the case when a SELECT statement has a column that is not in the GROUP BY clause.

For example, the following queries are processed server side:

```
SELECT Component, Timeendpos as Timeend FROM [UploadedModel] WHERE
Component = 'Saved' OR DEST_CITY_MARKET_ID != '' AND DEST_AIRPORT_ID NOT
IN ('1', '2') ORDER BY ORIGIN_AIRPORT_ID DESC LIMIT 5
SELECT AVG(DEST_AIRPORT_ID), ORIGIN_AIRPORT_ID FROM [UploadedModel]
GROUP BY ORIGIN_AIRPORT_ID HAVING AVG(DEST_AIRPORT_ID) > 0
```

Columns

Name	Type	Description
_time	<i>Datetime</i>	
DEST_AIRPORT_ID	<i>Int</i>	
DEST_AIRPORT_SEQ_ID	<i>Int</i>	
DEST_CITY_MARKET_ID	<i>Int</i>	
host	<i>String</i>	
linecount	<i>Int</i>	

ORIGIN_AIRPORT_ID	<i>Int</i>
ORIGIN_AIRPORT_SEQ_ID	<i>Int</i>
ORIGIN_CITY_MARKET_ID	<i>Int</i>
punct	<i>String</i>
source	<i>String</i>
sourcetype	<i>String</i>
splunk_server	<i>String</i>
timestamp	<i>String</i>

Stored Procedures

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

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

Splunk Adapter Stored Procedures

Name	Description
CreateSchema	Creates a schema file for the specified table or view.

CreateSchema

Creates a schema file for the specified table or view.

CreateSchema

Creates a local schema file (.rsd) from an existing table or view in the data model.

The schema file is created in the directory set in the Location connection property when this procedure is executed. You can edit the file to include or exclude columns, rename columns, or adjust column datatypes.

The adapter checks the Location to determine if the names of any .rsd files match a table or view in the data model. If there is a duplicate, the schema file will take precedence over the default instance of this table in the data model. If a schema file is present in Location that does not match an existing table or view, a new table or view entry is added to the data model of the adapter.

Input

Name	Type	Required	Accepts Output Streams	Description
TableName	String	True	False	The name of the table or view.
FileName	String	False	False	The full file path and name of the schema to generate. The Location connection property specifies the default output directory.
FileStream	String	False	True	An instance of an output stream where file data is written to. Only used if FileName is not set.

Result Set Columns

Name	Type	Description
Result	<i>String</i>	Returns Success or Failure.
FileData	<i>String</i>	If the FileName input is empty.

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
AuthScheme	Whether to use Basic Authentication or AccessToken Authentication when connecting to Splunk.
AccessToken	The Access Token used for accessing your Splunk account.
URL	The URL to your Splunk endpoint.
User	The Splunk user account used to authenticate.
Password	The password used to authenticate the user.

SSL

Property	Description
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
IncludeInternalFields	Whether or not the CData ADO.NET Provider for Splunk should push the internal fields. These fields include: user, eventtype, etc.
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.

MaxThreads	Specifies the number of concurrent requests. Only used when UseJobs is true.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Splunk.
Readonly	You can use this property to enforce read-only access to Splunk from the provider.
RowScanDepth	Set this property to control the number of rows scanned when TypeDetectionScheme is set to RowScan.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data type of columns.
UseJobs	Specifies whether to use the jobs endpoint instead of the export endpoint. If set to true, the maximum number of returned rows is configured Splunk's limit.conf file.
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
AuthScheme	Whether to use Basic Authentication or AccessToken Authentication when connecting to Splunk.
AccessToken	The Access Token used for accessing your Splunk account.

URL	The URL to your Splunk endpoint.
User	The Splunk user account used to authenticate.
Password	The password used to authenticate the user.

AuthScheme

Whether to use Basic Authentication or AccessToken Authentication when connecting to Splunk.

Possible Values

Basic, AccessToken

Data Type

string

Default Value

"Basic"

Remarks

- Basic: Set this to perform Basic authentication.
- AccessToken: Set this to perform Token Based Authentication via the [AccessToken](#) property.

AccessToken

The Access Token used for accessing your Splunk account.

Data Type

string

Default Value

""

Remarks

The Access Token used for accessing your Splunk account.

URL

The URL to your Splunk endpoint.

Data Type

string

Default Value

""

Remarks

The URL to your Splunk endpoint; for example, <https://yoursitename.splunk.com:8089>.

The port should be set to the Splunk management port (default 8089).

User

The Splunk user account used to authenticate.

Data Type

string

Default Value

""

Remarks

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

SSL

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

Property	Description
SSLServerCert	The certificate to be accepted from the server when connecting using TLS/SSL.

SSLServerCert

The certificate to be accepted from the server when connecting using TLS/SSL.

Data Type

string

Default Value

""

Remarks

If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine is rejected.

This property can take the following forms:

Description	Example
A full PEM Certificate (example shortened for brevity)	-----BEGIN CERTIFICATE----- MIICHTCCAe4CAQAwDQYJKoZIhvd.....Qw == -----END CERTIFICATE-----
A path to a local file containing the certificate	C:\cert.cer
The public key (example shortened for brevity)	-----BEGIN RSA PUBLIC KEY----- MIGfMA0GCSq.....AQAB -----END RSA PUBLIC KEY-----
The MD5 Thumbprint (hex values can also be either space or colon separated)	34e929226ae0819f2ec14b4a3d904f801c
The SHA1 Thumbprint (hex values can also be either space or colon separated)	bb150d

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

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

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

Firewall

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

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

FirewallType

The protocol used by a proxy-based firewall.

Possible Values

NONE, TUNNEL, SOCKS4, SOCKS5

Data Type

string

Default Value

"NONE"

Remarks

This property specifies the protocol that the adapter will use to tunnel traffic through the [FirewallServer](#) proxy. Note that by default, the adapter connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set [ProxyAutoDetect](#) to false.

Type	Default Port	Description
TUNNEL	80	When this is set, the adapter opens a connection to Splunk 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
<code>ProxyAutoDetect</code>	This indicates whether to use the system proxy settings or not. This takes precedence over other proxy settings, so you'll need to set <code>ProxyAutoDetect</code> to <code>FALSE</code> in order use custom proxy settings.
<code>ProxyServer</code>	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\\Splunk 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\\Splunk 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
IncludeInternalFields	Whether or not the CData ADO.NET Provider for Splunk should push the internal fields. These fields include: user, eventtype, etc.
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.
MaxThreads	Specifies the number of concurrent requests. Only used when UseJobs is true.
Other	These hidden properties are used only in specific use cases.
Pagesize	The maximum number of results to return per page from Splunk.
Readonly	You can use this property to enforce read-only access to Splunk from the provider.
RowScanDepth	Set this property to control the number of rows scanned when TypeDetectionScheme is set to RowScan.
Timeout	The value in seconds until the timeout error is thrown, canceling the operation.
TypeDetectionScheme	Determines how to determine the data type of columns.
UseJobs	Specifies whether to use the jobs endpoint instead of the export endpoint. If set to true, the maximum number of returned rows is configured Splunk's limit.conf file.
UserDefinedViews	A filepath pointing to the JSON configuration file containing your custom views.

IncludeInternalFields

Whether or not the CData ADO.NET Provider for Splunk should push the internal fields. These fields include: user, eventtype, etc.

Data Type

bool

Default Value

false

Remarks

Whether or not the Splunk Adapter should push the internal fields. These fields include: user, eventtype, etc.

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.

MaxThreads

Specifies the number of concurrent requests. Only used when UseJobs is true.

Data Type

string

Default Value

"5"

Remarks

This property allows you to issue multiple requests simultaneously, thereby improving performance. Default value is 5 threads. Setting a higher value can result in OutOfMemory issues.

Other

These hidden properties are used only in specific use cases.

Data Type

string

Default Value

""

Remarks

The properties listed below are available for specific use cases. Normal driver use cases and functionality should not require these properties.

Specify multiple properties in a semicolon-separated list.

Integration and Formatting

DefaultColumnSize	Sets the default length of string fields when the data source does not provide column length in the metadata. The default value is 2000.
ConvertDateTimeToGMT	Determines whether to convert date-time values to GMT, instead of the local time of the machine.
RecordToFile=filename	Records the underlying socket data transfer to the specified file.

Pagesize

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

Data Type

int

Default Value

10000

Remarks

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

Readonly

You can use this property to enforce read-only access to Splunk from the provider.

Data Type

bool

Default Value

false

Remarks

If this property is set to true, the adapter will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

RowScanDepth

Set this property to control the number of rows scanned when TypeDetectionScheme is set to RowScan.

Data Type

string

Default Value

"50"

Remarks

Determines the number of rows used to determine the column data types.

Setting a high value may decrease performance. Setting a low value may prevent the data type from being determined properly, especially when there is null data.

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.

TypeDetectionScheme

Determines how to determine the data type of columns.

Possible Values

None, RowScan

Data Type

string

Default Value

"RowScan"

Remarks

None	Setting TypeDetectionScheme to None will return all columns as the string type.
RowScan	Setting TypeDetectionScheme to RowScan will scan rows to heuristically determine the data type. The RowScanDepth determines the number of rows to be scanned.

UseJobs

Specifies whether to use the jobs endpoint instead of the export endpoint. If set to true, the maximum number of returned rows is configured Splunk's limit.conf file.

Data Type

bool

Default Value

false

Remarks

Whether to use the jobs endpoint instead of the export endpoint. While Jobs generally provide higher performance, the initial response time may be longer. If a Timeout error occurs, set the [Timeout](#) connection property to a higher value.

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 DataModels 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.