Command Reference: WriteTableToDataStore()

Write a table to a datastore

Version 10.18.00, 2013-03-03

This command is under development and has the following limitations:

- Although some error handling has been implemented, it is not very detailed. Improvements will be made in response to exercising the command functionality.
- Write statements are created for each row of the table being written. This is inefficient and slow. Improvements will be made in future updates.
- Functionality has been tested mainly with SQL Server.
- Handling of date objects has not been tested.
- Better handling of blank rows needs to be implemented.

The `WriteTableToDataStore()` command processes each row in a table and executes an SQL statement to insert the row into a database datastore. If database datastore support is not specifically provided by TSTool, a generic datastore can be used (see the **Generic Database Datastore** appendix). This command cannot be used with web service datastores and use with Excel datastores has not been tested. This command is useful in particular for bulk data loading such as for database initialization and when tight integration with TSTool is not required or has not been implemented. In the future additional command parameters may be added to limit the rows that are being written and allow update functionality.

General constraints on the query are as follows:

- the table or views being written must be writeable by the user specified for the database connection (some databases restrict direct access to data and require using stored procedures)
- the table column names must match the database table column names (in the future a command parameter may be added to allow column names to be mapped)
- data types for table columns must closely match the database:
  - internally an SQL statement is created in which data values are formatted as per the data type (e.g., strings are quoted); consequently column types must be appropriate to generate correct formatting
  - the full precision of floating point numbers is passed to the database (formatting for display will not apply to values written to the database)
  - null values in the table will transfer to null values in the database
  - date/time columns in the table will be represented as such in the database table; however, it may not be possible to limit the precision of the date/time (i.e., hours, minutes, and seconds may be shown with default zero values in output)
- the specified table columns are written (all are written by default)
  - primary keys in the database table do not need to be specified (their values will be assigned automatically)
  - table columns that correspond to related tables in the datastore table need to be mapped using the `DataStoreRelatedColumnsMap` command parameter

An example of column mapping to a related table is as follows, using the notation `Table.Column` to fully identify columns:

- the string `TableID.DataType` column is in the input data

- an integer database table `TimeSeriesMeta.DataTypesID` column is a foreign key to `DataTypes.DataTypesID`, and `DataTypes.Abbreviation` is the string data type – in other words, the datastore column being written does not match the string data type, but uses a relationship to match the integer data type in a separate table

To handle this relationship:

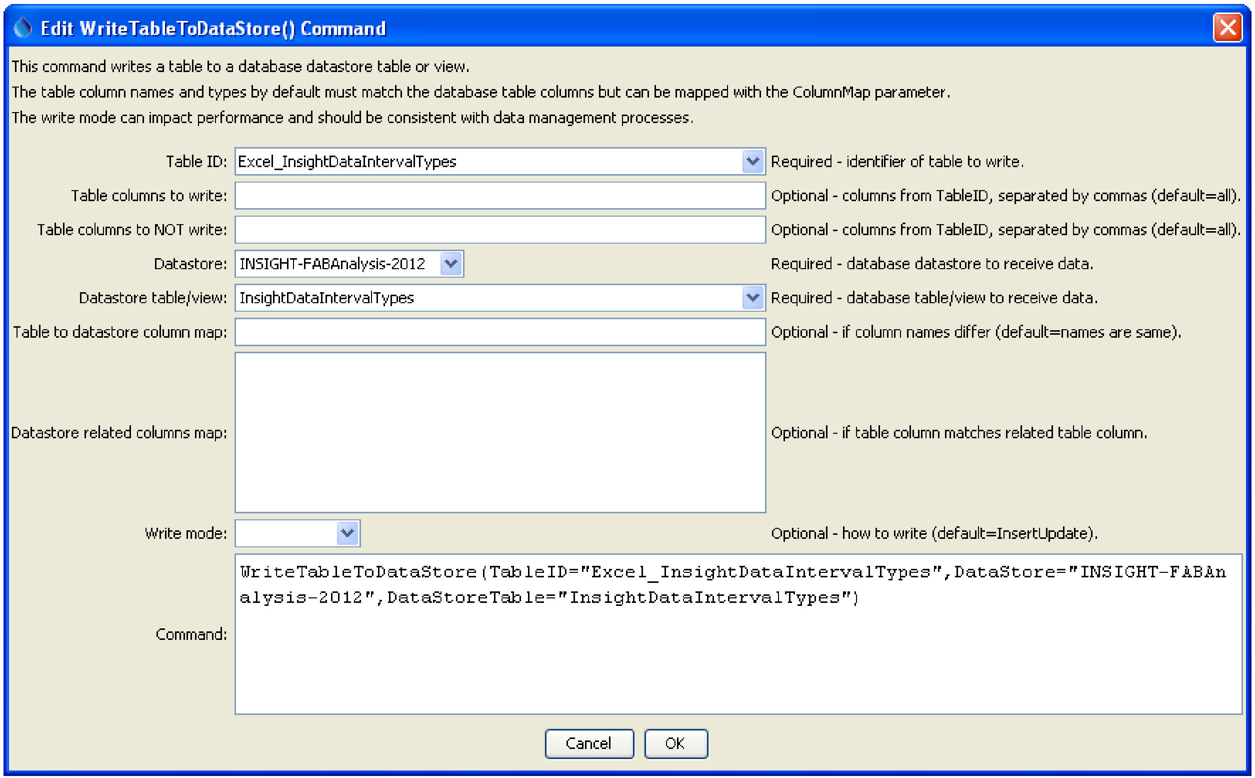
- Use the `ColumnMap` parameter to tell the command that the `DataType` column in input table maps to the `DataTypesID` column in the datastore table:

```
ColumnMap=”DataType:DataTypesID”
```

- Use the `DataStoreRelatedColumnsMap` parameter to tell the command that the `DataTypesID` column should be looked up the Abbreviation column, which is a second level of column mapping:

```
DataStoreRelatedColumnsMap=”DataTypesID:Abbreviation”
```

The following dialog is used to edit the command and illustrates the syntax for the command, in this case writing a table to a datastore that was defined as a GenericDatabaseDataStore.



This command writes a table to a database datastore table or view.
The table column names and types by default must match the database table columns but can be mapped with the ColumnMap parameter.
The write mode can impact performance and should be consistent with data management processes.

Table ID: Excel_InsightDataIntervalTypes — Required - identifier of table to write.
Table columns to write: — Optional - columns from TableID, separated by commas (default=all).
Table columns to NOT write: — Optional - columns from TableID, separated by commas (default=all).
Datastore: INSIGHT-FABAnalysis-2012 — Required - database datastore to receive data.
Datastore table/view: InsightDataIntervalTypes — Required - database table/view to receive data.
Table to datastore column map: — Optional - if column names differ (default=names are same).
Datastore related columns map: — Optional - if table column matches related table column.
Write mode: — Optional - how to write (default=InsertUpdate).
Command:

```
WriteTableToDataStore(TableID="Excel_InsightDataIntervalTypes",DataStore="INSIGHT-FABAn
alysis-2012",DataStoreTable="InsightDataIntervalTypes")
```

WriteTableToDataStore

WriteTableToDataStore() Command

The command syntax is as follows:

```
WriteTableToDataStore(Parameter=Value,…)
```

Command Parameters

Parameter	Description	Default
`TableID`	Identifier for table to write.	None – must be specified.
`IncludeColumns`	The names of the table columns to write, separated by commas.	All columns from `TableID` are written.
`ExcludeColumns`	The names of table columns NOT to write, separated by commas. This will override `IncludeColumns`.	All columns from `TableID` are written.
`DataStore`	The name of a database datastore to receive data.	None – must be specified.
`DataStoreTable`	The name of the database table or view to receive data.	None – must be specified.
`ColumnMap`	Indicate which columns in `TableID` have different names in `DataStoreTable`, using the syntax: `ColumnName:DatastoreTableName,` `ColumnName:DatastoreTableName,` …	`DatastoreTableName` columns are assumed to match the column names in `TableID`

`DataStoreRelatedColumnsMap`	Indicate datastore columns that need to match values in a related table in the datastore. For example, `TableID` may contain a column "Abbreviation" but the corresponding column in `DataStoreTable` may refer to a related table using a foreign key relationship (matching integer column in both tables). It is expected that the related table will have only one primary key column, which will be determined automatically. However, a column mapping must be provided to tell the command which `DataStoreTable` column should be matched with the related table. The syntax of the parameter is:
	`DataStoreTableCol1:RelatedTableCol1,`
	`DataStoreTableCol2:RelatedTableCol2,`
	…
	The above assumes that foreign keys have been defined in the `DataStoreTable` columns. If the database does not explicitly define a foreign key relationship in the database design, then specify the right side of the map as: `RelatedTable1.RelatedCol1`.
		`DatastoreTableName` columns are assumed to match the column names in `TableID`, with no need to perform reference table value matching.

`WriteMode`	The method used to write data, recognizing the databases use insert and update SQL statements, one of: • `DeleteInsert` – delete the data first and then insert (all values will need to be matched to delete) • `Insert` – insert the data with no	`InsertUpdate`

Parameter	Description	Default
	attempt to update if the insert fails • InsertUpdate – try inserting the data first and if that fails try to update • Update – update the data with no attempt to insert if the update fails • UpdateInsert – try updating the data first and if that fails try to insert	

This page is intentionally blank.