Command Reference: WriteTableToDataStore()

Write a table to a datastore

Version 10.18.00, 2013-02-26

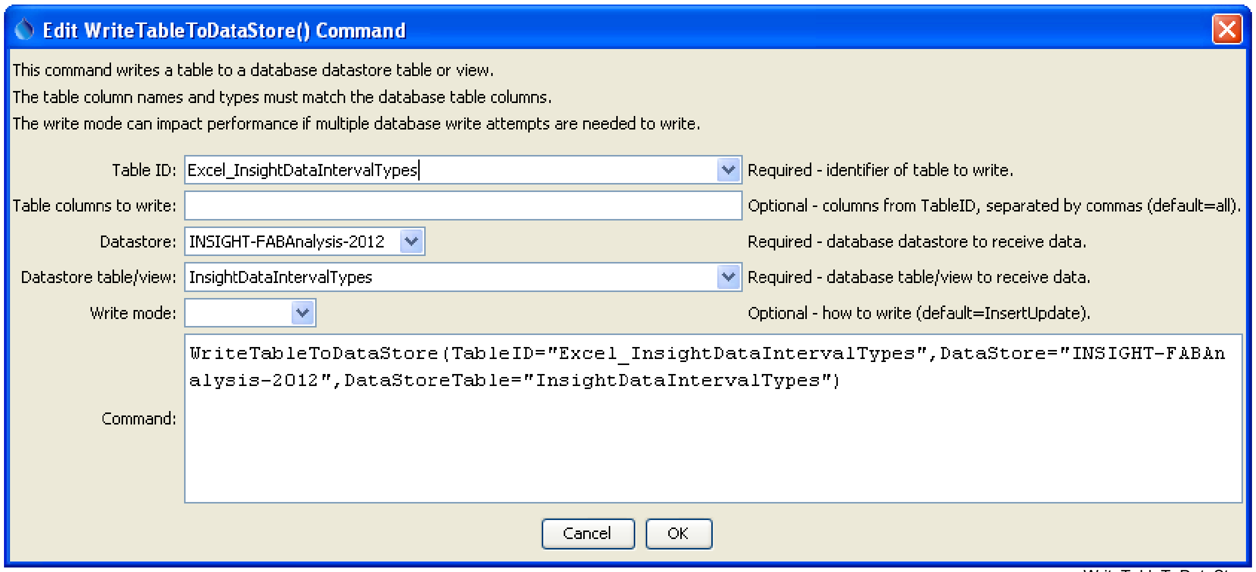
The `WriteTableToDataStore()` command processes each row in a table and executes an SQL statement to insert the row into a database datastore. If database datastore support is not specifically provided by TSTool, a generic datastore can be used (see the **Generic Database Datastore** appendix). This 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 automatically will be assigned)
    - foreign keys???

Future enhancements will add additional features to intelligently map TSTool tables into database tables.

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 must match the database table columns.
The write mode can impact performance if multiple database write attempts are needed to write.

Table ID:	Excel_InsightDataIntervalTypes	Required - identifier of table to write.
Table columns to 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.
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.
`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.
`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 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	`InsertUpdate`