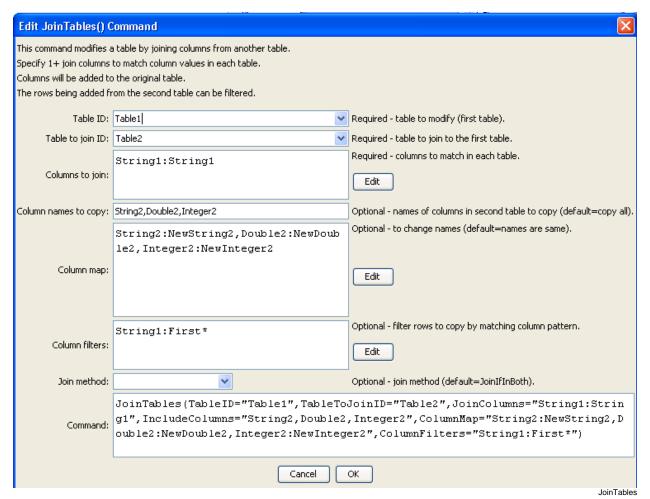# Command Reference: JoinTables()

Joins a second table to a first table by matching column values

Version 10.29.00, 2014-04-21

The `JoinTables()` command joins two tables that have matching column values, resulting in the first table being modified to contain additional columns from the second table. Rows from the first table are always retained and may optionally have new rows added. Depending on the join method, additional rows may be added in cases where one of the tables does not match the other. All or a subset of the columns and rows from the second table can be included in the result. The functionality of the command is similar to database SQL join statements using left, right, center, etc. join syntax. The following dialog is used to edit the command and illustrates the syntax of the command.

JoinTables() Command Editor

The command syntax is as follows:

```
JoinTables(Parameter=Value,…)
```

Command Parameters

Parameter	Description	Default
TableID	The identifier for the original table. This table will be modified.	None – must be

Parameter	Description	Default
		specified.
`TableToJoinID`	The identifier for the table to join.	None – must be specified.
`JoinColumns`	The names of columns to be compared in the join, using syntax: `Table1ColumnName1:Table2ColumnName1, Table1ColumnName2:Table2ColumnName2`	Required – must specify at least one column to compare
`Include Columns`	Specify the names of columns to copy from the second table into the first table, separated by commas. The columns indicated by `JoinColumns` will not be added because they should already be in the table.	Copy all of the columns from the `TableToJoinID` table.
`ColumnMap`	Specify new names for the output columns being included, using syntax: `OriginalTable2ColumnName1:NewColumnName1, OriginalTable2ColumnName2:NewColumnName2`	Column names in the result will be the same as in the original `TableToJoinID` table.
`Column Filters`	Filters that limit the number of rows being processed, using the syntax: `Table2Column1:FilterPattern1, Table2Column2:FilterPattern2` Patterns can use `*` to indicate wildcards for matches. Only string values can be checked (other data types are converted to strings for comparison). Comparisons are case-independent. All patterns must be matched in order to copy the row. In the future a command may be added to perform queries on tables, similar to SQL for databases.	No filtering.
`JoinMethod`	Indicate how the tables should be joined: • `JoinIfInBoth` – only add column values from the second table that have matching values in columns specified by `JoinColumns` • `JoinAlways` – add rows even if join column values do not match. Rows in the second table that don't match will result in new rows at the bottom of the table, with null/missing values for columns in the first table.	`JoinIfInBoth`
`Handle Duplicates InTableHow`	This parameter is not yet implemented. Indicate how to handle duplicate matches in the TableID table. The current default is that each record would be processed without warnings.	
`Handle Duplicates InJoin TableHow`	This parameter is not yet implemented. Indicate how to handle duplicate matches in the `TableToJoinID` table: • `DuplicateRows` – duplicate the original row for each match and join each row with one of the matched rows in the second table. This "flattens" the result by adding more rows. • `Fail` – generate a failure message and do not perform the join • `Warn` – generate a warning message and do not perform the join	`Fail`
`MatchColumn`	This parameter is not yet implemented. Indicate the name of a column to add to the joined output table to indicate how the join comparison evaluated. This is useful to help with "downstream" data processing. The value of the cell in the table will be set to one of: • `OnlyInFirst` • `OnlyInSecond` • `InBoth`	Do not add the column.