Command Reference: For()

Start a block of commands as part of a "for" loop

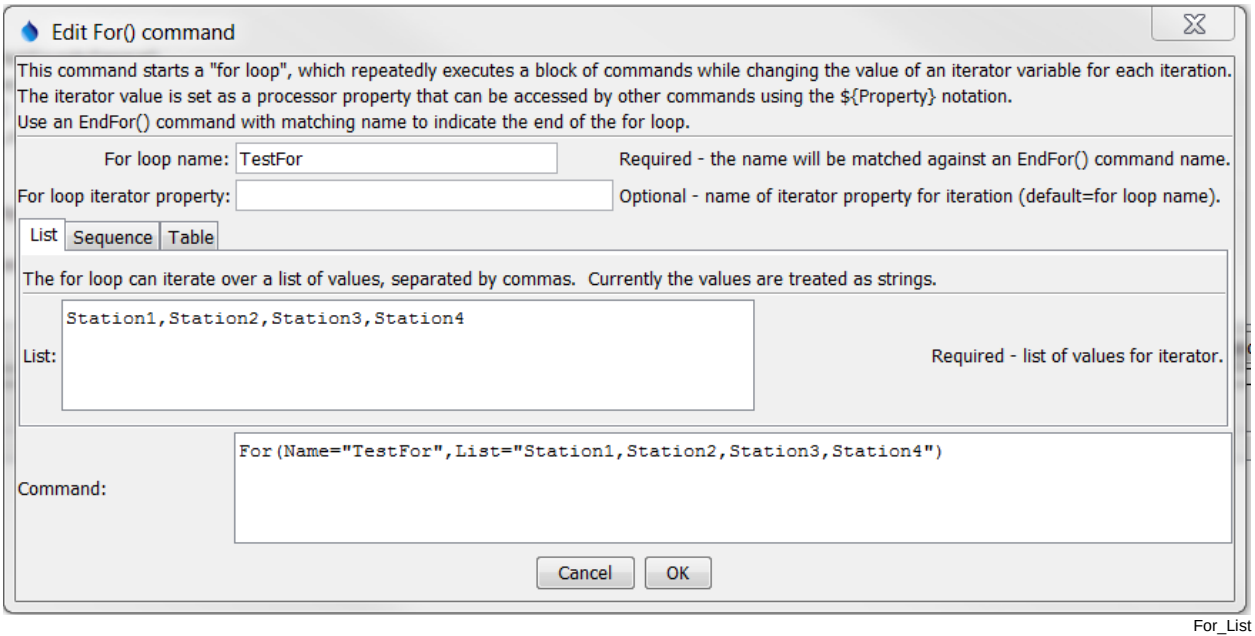
Version 11.10.00, 2016-04-09

The `For()` command iterates through a block of commands between `For()` and matching `EndFor()` commands. A processor property is set to the value of the iteration property and can be used by other commands that support properties, using the `${Property}` notation. This command is an alternative to implementing loops in templates (see `ExpandTemplateFile()`), in particular for straightforward command logic. `For()` commands can iterate over:

- a list of supplied values
- a sequence of integers or floating-point double precision numbers specified with start, end, and increment
- values from a table column

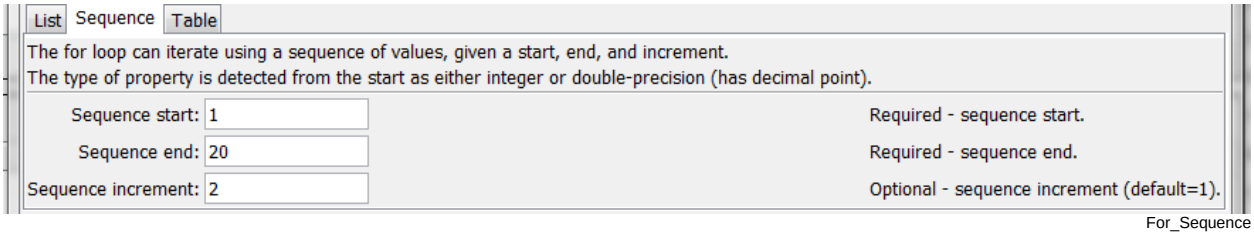
`For()` commands can be nested. Status messages for run mode are accumulated in each command (this update is occurring over time – status messages for some commands may be cleared out each iteration). A limitation of using `For()` with properties is that command when edited may show time series identifiers and other command parameters as `${Property}` values, rather than actual data, because the values get expanded at run-time. This provides increased processing power but errors may not be evident until commands re run.

The following dialog is used to edit this command and illustrates the command syntax.



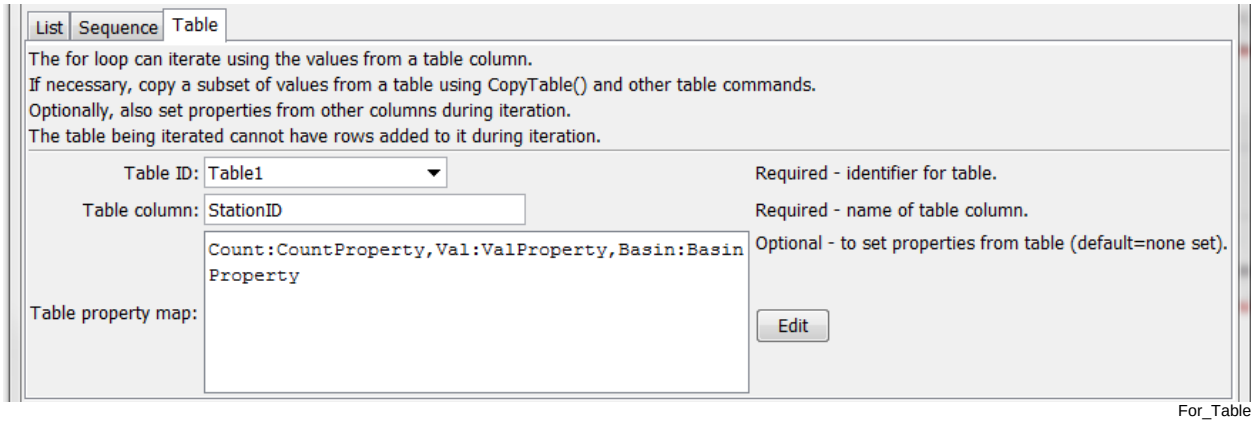
For_List

For() Command Editor Illustrating Using a List for Iteration Values



For_Sequence

For() Command Editor Illustrating Using a Sequence of Integers for Iteration Values



For_Table

For() Command Editor Illustrating Using a Table for Iteration Values

The command syntax is as follows:

```
For(Parameter=Value,…)
```

Command Parameters

Parameter	Description	Default
`Name`	The name of the "for" loop, which will be matched with the name of an `EndFor()` command to indicate the block of commands in the loop.	None – must be specified.
`IteratorProperty`	The processor property that will be set to the iterator property. The object type will depend on that used to provide the iteration property list. For example, if a column of strings from a table is used for iteration, the property will contain a string.	Same as `Name`.
`List`	A list of comma-separated values to be used as variables for the iteration.	None if list is used – must specified a list of values.
`SequenceStart`	Starting value when a sequence is specified for iteration, an integer or floating-point number (with decimal).	None if sequence is used.
`SequenceEnd`	Ending value for sequence.	None – must be specified if sequence is used.
`SequenceIncrement`	Increment for sequence iterator.	1 or 1.0 depending on `SequenceStart` type.

Parameter	Description	Default
TableID	The table identifier, when specifying the iterator as a column from a table. Can be specified with processor ${Property}.	None if table is used – must specify the table ID.
TableColumn	The table column name, when specifying the iterator as a column from a table.	None – must be specified if table is used.
TablePropertyMap	Specify the names of column names and corresponding processor property names to set. This allows other commands to access the values of those properties using ${Property} notation. Specify using format: ColumnName1:PropertyName1, ColumnName2:PropertyName2	None – only the iterator column value will be set as a property using IteratorProperty.

The following example illustrates a simple For() and EndFor() usage. In this example the StationID column in the input table is used to provide the list of values to iterate over. The following input table is a delimited file but could come from another source:

```
# Test table data for For() command tests
"Count","Val","StationID","Basin"
1,1.0,Station1,Basin1
2,2.0,Station2,Basin2
3,3.0,Station3,Basin3
4,4.0,Station4,Basin4
```

The following command file reads the above input table, iterates over the StationID column, and creates a simple output file:

```
ReadTableFromDelimitedFile(TableID="Table1",InputFile="Data\testtable.csv")
RemoveFile(InputFile="Results/Test_For_TableString_out.txt",IfNotFound=Ignore)
For(Name="TestFor",TableID="Table1",TableColumn="StationID")
WritePropertiesToFile(OutputFile="Results/Test_For_TableString_out.txt",
    IncludeProperty="TestFor",WriteMode=Append,FileFormat=NameTypeValue)
EndFor(Name="TestFor")
```

The resulting output file is as follows:

```
TestFor="Station1"
TestFor="Station2"
TestFor="Station3"
TestFor="Station4"
```

This page is intentionally blank.