# Command Reference: If()

## Start a block of commands as part of a conditional "if"

Version 12.00.00, 2017-03-25

The `If()` command evaluates a conditional statement and if true will result in the commands between `If()` and matching `EndIf()` being executed. Matching `Name` parameter for the `If()` and `EndIf()` commands defines a block of commands. Currently, there is no "else if" or "else" syntax and nested `If()` commands must be used to evaluate complex conditions. The syntax for the conditional statement is restricted to a simple comparison:

Value1 operator Value2

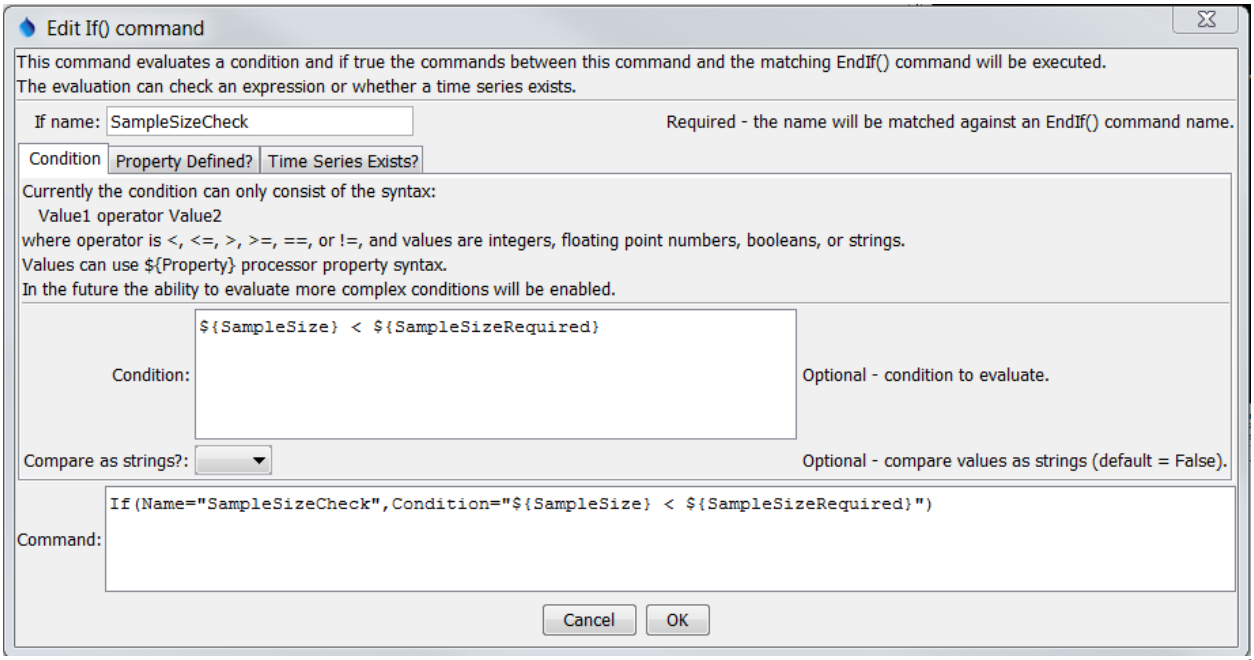
The values can be integers, floating point numbers, Boolean values, strings, or processor properties that evaluate to these types. The operator is one of the following (more functionality will be added in the future). For Booleans, `False` is less than `True`. For strings, `A` is less than `Z`, etc.

- <
- <=
- >
- >=
- == (use this to test equality – do not use a single equal sign)
- !=

All nested `If()` commands must evaluate to true to execute the commands within the deepest level of nesting. Some commands, including `SelectTimeSeries`() and `CopyTable()` set a property that can be used for checks, for example to see if the number of rows in a table or time series in a list is non-zero.

It is helpful to use the `Message()` command to print a message to the log file and help control whether a warning or failure status should occur as the result of the `If()`.

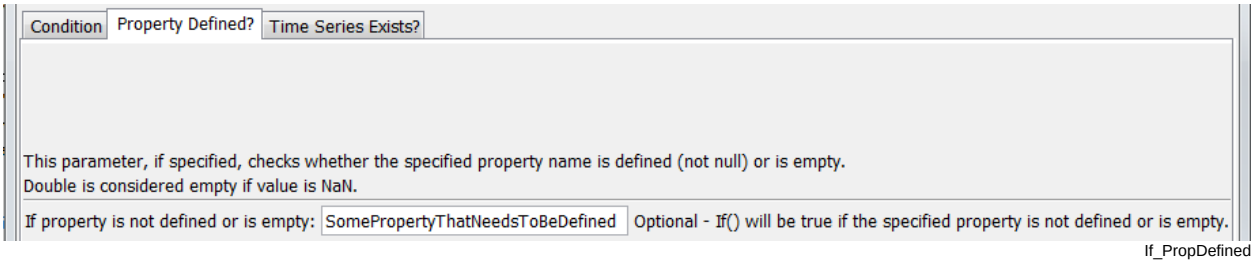
The following dialog is used to edit this command and illustrates the command syntax.




If

**If() Command Editor Showing Condition to Test**

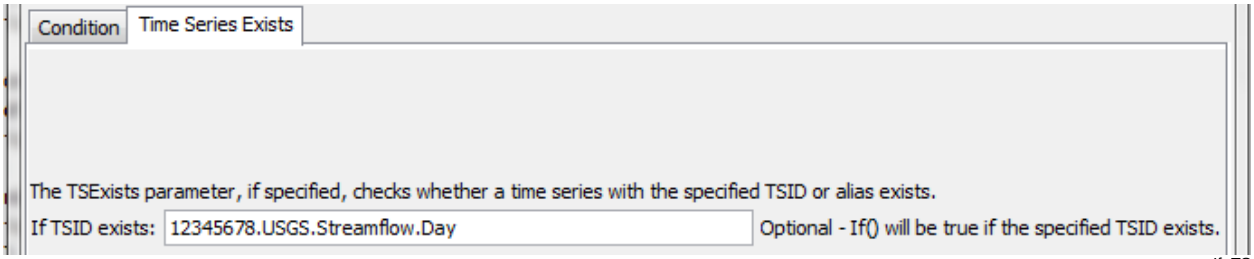
The following illustrates checking for a property to make sure it is defined and not empty. This is useful for detecting logic and data problems.




If_PropDefined

**If() Command Editor Showing Check for Whether a Property is Not Defined or is Empty**

The following illustrates how to detect if a time series exists. This is useful for executing only blocks of commands that operate on the time series (and avoiding those steps and related warning/failure messages when the time series does not exist).




If_TS

**If() Command Editor Showing Check for Time Series Existence**

The command syntax is as follows:

```
If(Parameter=Value,…)
```

**Command Parameters**

| Parameter | Description | Default |
|---|---|---|
| `Name` | The name of the "if" command, which will be matched with the name of an `EndIf()` command to indicate the block of commands in the "if" command. | None – must be specified. |
| `Condition` | The conditional statement to evaluate. | `Condition` and/or `TSExists` and/or `PropertyIsNotDefinedOrIsEmpty` must be specified. |
| `CompareAsStrings` | If `True`, the comparison will be done as strings even if the values could be treated as numbers or Booleans. | `False` |
| `Property IsNotDefined OrIsEmpty` | Causes the command to evaluate to True if the specified parameter (a property name) is not defined or has a value of null, `NaN` (floating point numbers), or is an empty string. | `PropertyIsNotDefinedOrIsEmpty` and/or `Condition` and/or `TSExists` must be specified. |
| `TSExists` | Causes the command to evaluate to True if the specified time series does exist. Specify a TSID or alias to match. Can specify using `${Property}`. | `Condition` and/or `TSExists` and/or `PropertyIsNotDefinedOrIsEmpty` must be specified. |

The following example illustrates combinations of `If()` and `Message()` commands (indentation indicates line continuation). In these examples processor properties are used to provide condition values.

```
# Test evaluating an integer condition where integer is supplied by property
StartLog(LogFile="Results/Test_If_IntegerProperty_LT_IntegerProperty.TSTool.log")
SetProperty(PropertyName="SampleSizeRequired",PropertyType=Integer,PropertyValue="10")
SetProperty(PropertyName="SampleSize",PropertyType=Integer,PropertyValue="5")
If(Name="SampleSizeCheck",Condition="${SampleSize} < ${SampleSizeRequired}")
Message(Message="Sample size (${SampleSize}) is less than required
  ${SampleSizeRequired}",CommandStatus=WARNING)
EndIf(Name="SampleSizeCheck")
If(Name="SampleSizeCheck2",Condition="${SampleSize} > ${SampleSizeRequired}")
Message(Message="Sample size (${SampleSize}) is >= than required
  ${SampleSizeRequired}",CommandStatus=WARNING)
EndIf(Name="SampleSizeCheck2")
If(Name="SampleSizeCheck3Outer",Condition="${SampleSize} < ${SampleSizeRequired}")
If(Name="SampleSizeCheck3InnerTrue",Condition="${SampleSize} == 5")
Message(Message="Sample size (${SampleSize}) is == 5",CommandStatus=WARNING)
EndIf(Name="SampleSizeCheck3InnerTrue")
If(Name="SampleSizeCheck3InnerFalse",Condition="${SampleSize} != 6")
Message(Message="Sample size (${SampleSize}) is not == 6",CommandStatus=WARNING)
EndIf(Name="SampleSizeCheck3InnerFalse")
EndIf(Name="SampleSizeCheck3Outer")
```