Implementation Specifications or Requirements

Category	Item	
Software	POV Version:	7.1 and later
	Service Pack:	N/A
	Windows NT/2000/XP/7/8:	Yes
	Web Thin Client:	N/A
Equipment	Panel Manufacturer:	N/A
	Panel Model:	N/A
	Other Hardware:	N/A
	Comm. Driver:	All
	Controller (e.g.: PLC)	All
	Application Language:	N/A
Software Demo Application	N/A	

Summary

This Application Note describes the functioning of the Studio HST Server COM Object, which allows any COM handling enable scripting language (such as VBScript) to query data from the Studio trend history binary files.

Installing the StudioHstServer COM object

The Studio HST Server object is installed automatically when installing POV version 7.1 SP2 or higher.

However, if you need to manually install the Studio HST Server object, you need to have the **StudioHstServer.dll** and register it (for example, using the command ***RegSvr32 <File Path>\StudioHstServer.dll***).

It can be installed anywhere on your computer, although we recommend that you do it on the Studio folder, in order to keep track of the file

StudioHSTServer object name on Visual Basic

When calling the StudioHstServer on VisualBasic, you have different approaches for Visual Basic (VB), Visual Basic for Applications (VBA) and Visual Basic Script (VBS)

- On VB and VBA:

```
Dim HSTvar As StudioHstLib.StudioHst
Set HSTvar = New StudioHst
```

- On VBS

```
Dim HSTvar
Set HSTvar = CreateObject("StudioHst.HstMgt")
```

Properties and Methods

You can access different properties and methods of this Studio HST Server COM object.

Some of the Properties are inputs to run a Method. Other properties are actually return values after executing a Method

This object has only 2 Method: one to execute the query and one to move within the query.

In order to execute the Query, you need to enter the initial parameters in some of the properties.

After executing the Query, the return values are loaded in some of the properties that are described below.

Methods

`ExecuteFilter(start, end, bounds as Boolean)`

This is the Method that performs the Query of data. The parameters are

- **Start date and time**, date coming first, where the year requires 4 digits, separating date from time using a space and the time in the 24 hours format.
  Example: "30/04/2007 16:00:00"
- **End date and time** (same format as above)
- **Bounds:** If this parameter is False, the method will get the data within the Date/Time specified as a closed interval (values equal to start/end time will be returned). If the parameter is True and there is no value that matches the start time or the end time, the method will retrieve the first before the start time and the first after the end time

Before executing this function, you need at least to load the `hstPath` and `Group` properties.
This Method will return **True** if there is data in the given interval, group and path. Otherwise, it will return **False**

`MoveNext()`

Once you execute the `ExecuteFilter` method, it will point to the first record. The `MoveNext` method will move the pointer to the next record.
This Method will return **True** if there is data in the given interval, group and path. Otherwise, it will return **False**

Properties

INPUTs:

`hstPath`
: String value with Path where the trend history files are located. Usually on the application \HST folder. Usually you can load on this property the result of this expression: *$GetAppPath() + "hst"*

`Group`
: Numerical value with the Trend worksheet number that has the tags to be accessed

OUTPUTs:

`TagName`
: Array that once the `ExecuteFilter` method is executed, it will receive the list of the Tag Names from that Trend Group

`TagIndex`
: When calling this property you need to pass the tag name and it will return what is the numerical Index for that Tag Name. For example: `Hst.TagIndex("TagCos")` will return 0

`TimeStamp`
: Once the `ExecuteFilter` method is executed, this property will receive the TimeStamp from the first record. After executing the `MoveNext` Method, this property will receive of the next record

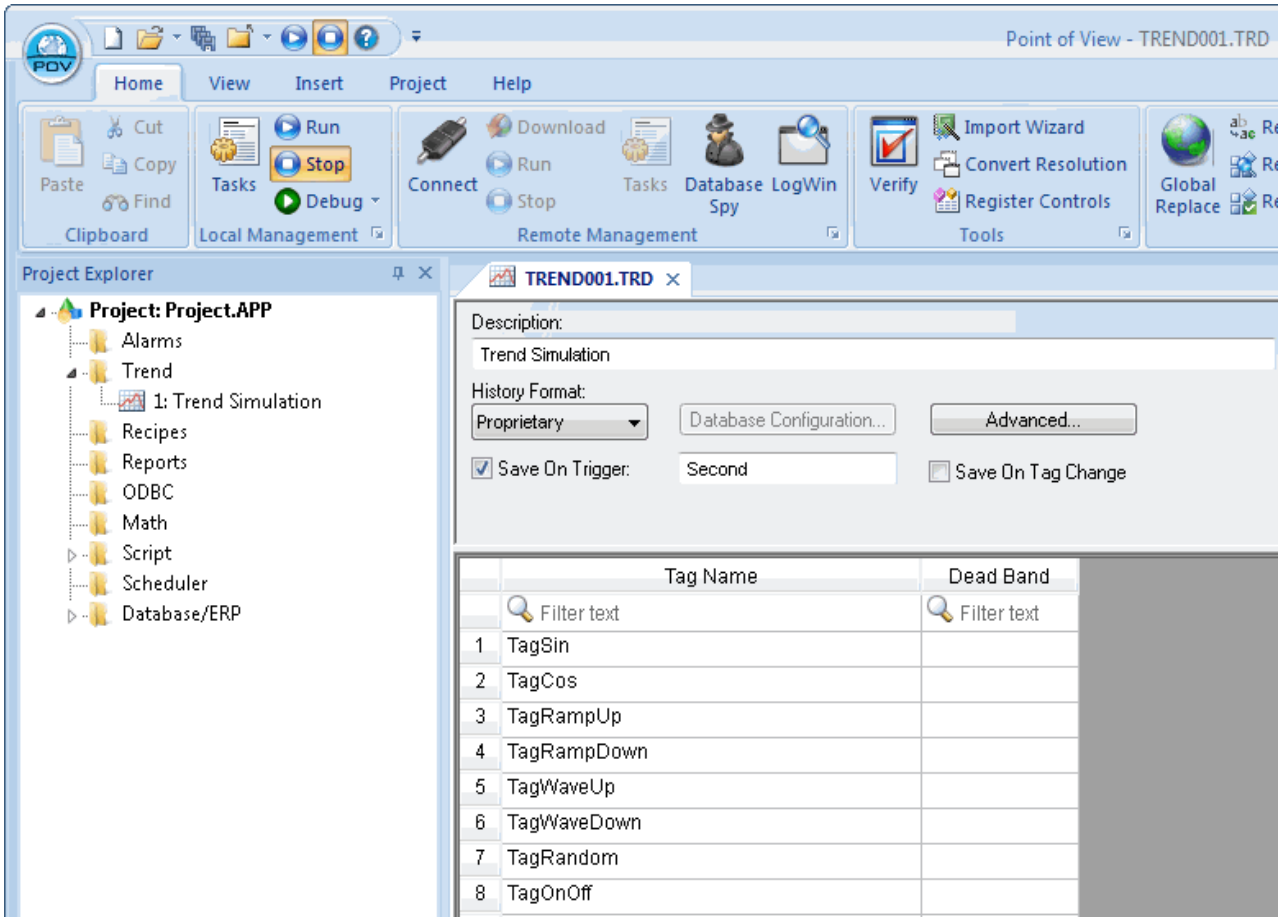
`TagValue`
: Array that once the `ExecuteFilter` method is executed will receive the Values for the tags.

Example

In this example, we will write a Studio-based VBScript code that will export the trend values from the default Studio Demo application, based on a given start and end date/time.

- **Trend Worksheet:**

  The Trend Worksheet is configured as follows:



- **VBScript:**

  The Script below is going to take the advantage of some built-in Studio functions in order to obtain things like the trend history files path

  - **Creating the COM Object:**

    Using VBScript, you would create the COM object using the CreateObject("StudioHst.HstMgt") function

```
Dim HST
Set HST = CreateObject("StudioHst.HstMgt")
```

- **Assigning the Input Parameters:**

  There are properties that you need to enter values to: Group and HstPath

```
HST.Group = 1 'Trend Worksheet Number
HST.hstPath = $GetAppPath() & "\hst" 'Path for the Trend History filesDim HST
```

- **Entering Start and End Date/Time:**

  In order to execute the query, you need to pass the start and end date/time in Date format.

  You can concatenate strings in the format “MM/DD/YYYY HH:MM:SS”

  So, in this example, we will create **Start** and **Finish** variables, to receive the period for the Query.

```
Dim Start, Finish
Start = “09/15/2007 11:00:00”
Finish = “09/15/2007 17:00:00”
```

  Or, you could use tags with values entered on the application, like Start Date and Time from the Trend Object

```
Dim Start, Finish
Start = $TrendStart
Finish = $TrendFinish
```

- **Executing the Query**

  Now you can run the Query that will acquire the data from the history files putting them on the *HST* object

```
HST.ExecuteFilter Start,Finish,False
```

- **Understand the return values**

  Once you have executed the Query, the *Output* properties will have received the values for the TagNames, Indexes, TimeStamp and values.

  So, run the Method as shown below

```
HST.ExecuteFilter Start,Finish,False
```

  Acording to the configuration above, you should receive these values:

Variable	Value	Note
HST.TagName(0)	**TagSin**	Tag Name in the Trend Worksheet
HST.TagName(1)	**TagCos**	Tag Name in the Trend Worksheet
HST.TagName(2)	**TagRampUp**	Tag Name in the Trend Worksheet
HST.TagName(3)	**TagRampDown**	Tag Name in the Trend Worksheet
HST.TagName(4)	**TagWaveUp**	Tag Name in the Trend Worksheet
HST.TagName(5)	**TagWaveDown**	Tag Name in the Trend Worksheet
HST.TagName(6)	**TagRandom**	Tag Name in the Trend Worksheet
HST.TagName(7)	**TagOnOff**	Tag Name in the Trend Worksheet

Variable	Value	Note
HST.TagIndex("TagSin")	**0**	Tag Index in the TagName Array
HST.TagNames("TagCos")	**1**	Tag Index in the TagName Array
HST.TagNames("TagOnOff")	**7**	Tag Index in the TagName Array
HST.TagNames(HST.TagName(4))	**4**	Tag Index in the TagName Array

Variable	Value	Note
HST.TimeStamp	**9/15/2007 11:00:00 AM**	Timestamp of the first record

Variable	Value	Note
HST.TagValue(0)	**12.5333233564304**	Value for the Tag with the Index 0 (TagSin) on the first record
HST.TagValue(1)	**99.2114701314478**	Value for the Tag with the Index 1 (TagCos) on the first record
HST.TagValue(2)	**-96**	Value for the Tag with the Index 2 (TagRampUp) on the first record
HST.TagValue(3)	**96**	Value for the Tag with the Index 3 (TagRampDown) on the first record
HST.TagValue(4)	**1.90211303259031**	Value for the Tag with the Index 4 (TagWaveUp) on the first record
HST.TagValue(5)	**93.203538596925**	Value for the Tag with the Index 5 (TagWaveDown) on the first record
HST.TagValue(6)	**-14.3101290932951**	Value for the Tag with the Index 6 (TagRandom) on the first record
HST.TagValue(7)	**-50**	Value for the Tag with the Index 6 (TagOnOff) on the first record

- **Moving the COM Object Pointer to the next Record**

  In order to get the values from the next Record, you need to run the Method **MoveNext**

  HST.MoveNext

Sample POV VBScript

In this sample, we will write a POV-based VBScript code that will export the trend values from the default POV Demo application, on the current Date, from 8 AM to 5 PM.

```
'Variables available only for this group can be declared here.

Dim HST 'COM Object Name
Dim Start 'Start Date and Time
Dim Finish 'End Date and Time
Dim WriteStr 'String used to write to the CSV FIle
Dim Sep 'CSV File Separator

Sep = "," 'Makes comma as separator
'The code configured here is executed while the condition configured in the Execution field is TRUE.

Set HST = CreateObject("StudioHst.HstMgt") 'Creates the COM Object

HST.Group = 1 'Sets Group as Trend Group 1
HST.hstPath = $GetAppPath() & "\hst" 'Set's the Trend History file to the app \HST folder

Start = $Date & " 08:00:00" 'Sets Start Date and time to the current date, 8 AM
Finish = $Date & " 17:00:00" 'Sets End Date and time to the current date, 5 PM

HST.ExecuteFilter Start,Finish,False 'Executes the Query

'Prepares the CSV File Header
WriteStr = "Time Stamp" & Sep & HST.TagName(0) & Sep & HST.TagName(1)_
 & Sep & HST.TagName(2) & Sep & HST.TagName(3)_
  & Sep & HST.TagName(4) & Sep & HST.TagName(5)_
   & Sep & HST.TagName(6) & Sep & HST.TagName(7)

'Writes to the CSV File the Header
$FileWrite("c:\hst.csv",WriteStr,0)

Dim Cond 'Prepares the test for the MoveNext method
Cond = True

Do While Cond
'Prepares the string with the values
WriteStr = HST.TimeStamp & Sep & HST.TagValue(0) & Sep & HST.TagValue(1)_
 & Sep & HST.TagValue(2) & Sep & HST.TagValue(3)_
  & Sep & HST.TagValue(4) & Sep & HST.TagValue(5)_
   & Sep & HST.TagValue(6) & Sep & HST.TagValue(7)
'Appends the CSV file with values
$FileWrite("c:\hst.csv",WriteStr,1)

Cond = HST.MoveNext() 'Moves to the Next Record

Loop 'While MoveNext returns True, means that there are still records on the Query

Set HST = Nothing 'Once done with the Query, kills the COM Object
```