



GENESIS64

Using Extended Point Syntax



APPLICATION NOTE

September 2013

Description: Guide to using Extended Point Syntax in GENESIS64.

OS Requirement: Vista x64 / Windows Server 2008 x64 / Windows 7 x64 / Windows Server 2008 R2 x64 / Windows 8 x64 / Windows Server 2012

General Requirement:

Introduction

The GENESIS64 suite follows OPC specifications to access OPC server data. Typically the values provided by the OPC items are all that is necessary; however in some cases more information such as quality or timestamp of the value is needed. In such situation users may take advantage of Extended Point Syntax.

The first part of this Application Note will go over how to use the supported Extended Point Syntax that was inherited from GENESIS32 and applies to OPC DA tags. The second part will explain the new unique features available in the GENESIS64 v10 release which apply to both OPC DA and OPC UA tags.

Extended Point Syntax inherited from GENESIS32

All OPC DA items using the EPS (Extended Point Syntax) inherited from GENESIS32 suite will have to be prefixed by **@tag:** and suffixed by **.Value**. It is possible to manually enter the Extended Point Syntax for an OPC item in the PPT edit box or in the Unified Data Browser. There is a related application note called *GENESIS32 – Using Extended Point Syntax* which describes how to use EPS Editor in version 9.

Requested Data Type Override

In the majority of cases, an OPC item is requested using its native datatype. For this there is no special action required in any of the ICONICS applications. Nevertheless, in some cases it is useful to force a datatype for the OPC item.

The Extended Point Syntax supports the following range datatypes, using the names defined in the Variant Type enumerator:

- | | |
|---------------------------------|------------------------------|
| • I1 (8 bit signed integer) | • R4 (32 bit floating point) |
| • I2 (16 bit signed integer) | • R8 (64 bit floating point) |
| • I4 (32 bit signed integer) | • BOOL (boolean) |
| • I8 (64 bit signed integer) | • BSTR (string) |
| • UI1 (8 bit unsigned integer) | • DATE (date/time format) |
| • UI2 (16 bit unsigned integer) | • DECIMAL (decimal) |
| • UI4 (32 bit unsigned integer) | • HRESULT |
| • UI8 (64 bit unsigned integer) | • ERROR |

To change the requested datatype for the OPC item use the Extended Point Syntax and manually enter the following format:

```
@tag:OPCServer\OPCTag|DataType.Value.
```

Array Indices

When data servers provide array data it may be necessary to extract a specific element from the array, this can be achieved using the Array Indices of the Extended Point Syntax. It supports single and multiple dimensional arrays. To extract the element separate the index for each dimension with a comma (,).

To obtain specific elements from an OPC item use the Extended Point Syntax and manually enter the following format:

```
@tag:OPCServer\OPCTag[index].Value
or
@tag:OPCServer\OPCTag[index1,index2].Value.
```

Array elements are always read-only, and cannot be written to, only the array tag as a whole (without any indices) is writeable.

Attribute

When an OPC Server sends a callback with the most up to date value to the OPC client, it also sends two other bits of information, the quality and a timestamp. These attributes can be visualized using the Extended Point Syntax.

The attributes provided are:

- **Quality** - The exact quality and sub quality provided by the OPC Server. This attribute shows the value from the OPC Server, not the value internally determined in FrameWorX, this means that when the connection the OPC Server breaks, this attribute will still show the last received value from the OPC Server. To accommodate for this scenario use the quality() function of the expression engine instead.
- **Good quality** - A convenience value returning 0 or 1.
- **Timestamp** - The timestamp sent with the last callback.
- **Valid** - A 0 or 1 value indicating if the tag is valid.

To request a property use the Extended Point Syntax and manually enter the following format:

```
@tag:OPCServer\OPCTag#attribute.Value.
```

Parameter Combinations

The parameters may be combined together. This could, for example, be used to request the timestamp attribute and request the data type to be a string:



GENESIS64

Using Extended Point Syntax



APPLICATION NOTE

September 2013

@tag:OPCServer\OPCTag#Timestamp|BSTR.Value

Examples

If you want to try out the inherited EPS functionality in GENESIS64, you can simply create your own GraphWorX64 display with a couple of PPTs and connect to your OPC Server (e.g. Simulation data). Then GraphWorX64 allows you to display the property of the tag in runtime by using the Extended Point Syntax functionality.

Follow these steps to display the most useful property such as Quality or Timestamp:

1. Launch the GraphWorX64 by going to **Start → Programs → ICONICS → GENESIS64 → GraphWorX64 → GraphWorX64**.
2. Click the Dynamics tab and add a new **PPT** by clicking this icon .
3. Browse for an example for a simulation tag with the following name:
`@ICONICS.Simulator.1\SimulatePLC.Sine.Value`
or any other tag you will find in the Data Browser to get the real value.
4. Duplicate the PPT object and browse for the same tag again, but now you will modify the existing point syntax by clicking the Edit field at the top and adding special EPS syntax, as shown on Figure 1.

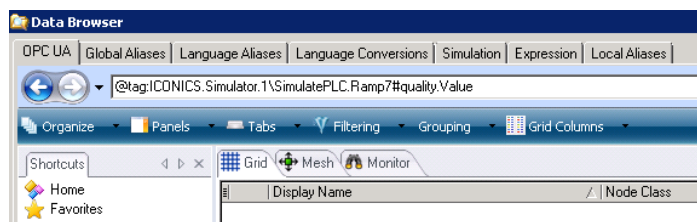


Figure 1 - Adding EPS Syntax Manually in Data Browser

5. Please make sure that you have the right syntax as it is shown on this example:
`@tag:ICONICS.Simulator.1\SimulatePLC.Sine#quality.Value`
6. When you add a couple of PPTs with different properties on a GraphWorX64 display you would get the similar output as it is shown on Figure 2.

@ICONICS.Simulator.1\SimulatePLC.Sine.Value 0.23

@tag:ICONICS.Simulator.1\SimulatePLC.Sine#quality.Value 192.00

@tag:ICONICS.Simulator.1\SimulatePLC.Sine#timestamp.Value 4/28/2010 11:29:32 AM

Figure 2 - EPS in GraphWorX64

OPC UA Extended Point Syntax introduced in GENESIS64

Since GENESIS64 v10.6 release there is the support for Extended Point Syntax with OPC UA tags, which is facilitated through the new EPS dropdown available in the Data Browser.

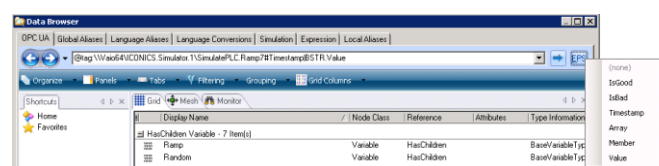


Figure 3 - EPS from Data Browser

NOTE: This syntax is also supported for OPC DA tags as shown in the examples below.

The following modifiers are available:

IsGood/IsBad – returns a Boolean value if the specified point is good/bad.

`isgood(@ICONICS.Simulator\SimulatePLC.Sine.Value)`
`isbad(@ICONICS.Simulator\SimulatePLC.Sine.Value)`

TimeStamp – returns a timestamp of the specified value.

`timestamplocal(@ICONICS.Simulator\SimulatePLC.Sine.Value)`
`timestamputc(@ICONICS.Simulator\SimulatePLC.Sine.Value)`
`timestamp(@ICONICS.Simulator\SimulatePLC.Sine.Value)`

Note: Use timestamplocal or timestamputc modifier if creating new expressions. Simple "Timestamp" is obsolete but kept for compatibility reason.

Array – returns specified element from an array value. The indices may or may not be enclosed in square brackets. Adding Length parameter at the end returns the array's length.

`array(\:Server.:NamespaceArray)[0]`
`array(\:Server.:NamespaceArray)0`
`array(\:Server.:NamespaceArray).Length`



GENESIS64

Using Extended Point Syntax



APPLICATION NOTE

September 2013

Member – returns the given member of a complex objects.

member(\\.\Server::ServerStatus).CurrentTime

Value – returns the actual value of the specified point. This may be useful in the scenario where a point starts with one of the other modifier keywords.

value(isgood()\\.\Server::ServerStatus)

Scan – this parameter allows to override scan rate of an OPC item. It works for both OPC DA and OPC UA points

- OPC UA:

*opc.tcp://T7:51210/UA/SampleServer -
[None:None]\\[http://test.org/UA/Data/]Data.Dynamic.Scalar.Int6
4Value(scan=750)*

- OPC DA:

@\\T7\ICONICS.Simulator.1\SimulatePLC.Sine.Value(scan=2500)

Note: The legacy syntax valid from GENESIS32 can be used for OPC DA points to override scan rate as well:

@tag:\\t7\ICONICS.Simulator.1\SimulatePLC.Ramp4@750.Value

Note: If the syntax of the given OPC point ends with parentheses “)” then use additional empty parentheses on the end of the name. For instance, if the original point name is “aPointName(with_parentheses)” then it would no longer work, because “with_parentheses” will be treated as a list of parameters (e.g. scan). The solution here is to use empty parentheses on the end. i.e. “aPointName(with_parentheses)()”.

Enum – converts numerical status information received from BACnet objects or Windows service from integer form to better human readable string notation.

*enum(bacnet:ICONICS BACnet Simulator 1000\AV 1\presentValue)
enum (:Services/AppIDSvc/Status)*

Note: The BACnet data translation is stored within the object itself if it is supported.

Format – Some BACnet objects expose a “resolution” property which determines the number of decimal places. The Format modifier will use this data when displaying the number and it will also display units after the value if they are specified in “units” property.

*format(bacnet:ICONICS BACnet Simulator 1000\AV
1\presentValue)*