Release Notes V4.4

07 05 22 cs

Table of Contents

Upgrading to Proview V4.4.0

This document describes new functions i Proview V4.4.0, and how to upgrade a project from V4.3.0 to V4.4.0.

New functions

OPC XML/DA Server

An OPC XML/DA Server is a web service from which an OPC XML/DA Client can get information of a Proview system. A opc client can, for example, browse the object hierachy, read and write attribute value, and subscribe to attributes.

The opc server implements the http protocol as well and is not connected to a web server. The port number of the opc_server is set to 80, the URI for the web service is on node 'mynode' is

`http://mynode`

If a web server is present, this normally has allocated the port 80, and another port has to be choosen for the opc_sever. If port 8080 is choosen, the URI will be

`http://mynode:8080`

Browsing

The OPC XML/DA browsing supports branches and items. The item contains a value, while the branch is an hierarchical component without a value. There is no support for objects, so an object has to be implemented as a branch, while an attribute is implemented as an item. Also arrays are implemented as branches, while array elements (that is not a class) is implemented as an item.

Threads

If the opc client uses the HoldTime and WaitTime attributes in the SubscriptionPolledRefresh request, the opc server has to be multi threaded, that is, for every request, an new thread is created. If the HoldTime and WaitTime is not used (as in the proview opc client), all requests can be handled in a single thread, which is less time consuming. Multithreads or not is configured in the configuration object for the opc server. The default value is 'IfNeeded' which turns on the multithreading for a client if HoldTime or WaitTime are detected.

Client access

To gain access to a proview opc server, the ip address of the client has to be configured in the configuration object for the opc server. Here you can also choose if the client has ReadWrite or ReadOnly access, where ReadOnly, allows the client to read and subscribe to attribute values, while ReadWrite also is allowed to write attribute values.

Buffering of subscriptions

The server does not support buffering of subscriptions.

Configuration

The opc server is configured with a Opc_ServerConfig object that is placed under in the Node hierarchy. The configuration object will cause a server process (opc_server) to start at proview startup.

OPC XML/DA Client

The proview opc client is implemented as an extern volume, which is mounted somewhere in the objects tree of the root volume. Under the mount object, the branches and items of the server are displayed with special opc objects. An Opc_Hier object represents a branch and Opc_Int an item with an integer value, Opc_Boolean an item with a boolean value etc. If an item object is opened the item value are displayed in a Value attribute, and some other properties as description, lowEU, highEU, engineeringUnit, lowIR and highIR are displayed. When the object is opened a subscription is started, and the value is continuously updated. For integer and float items there is also an object graph that displays a trend of the value.

With the opc client you can

- browse the branches and items in Xtt, and also display item values and set item values.
- subscribe item values and display them in a Ge graph.
- fetch item values into the plc logic and also write values to items.

The opc client requires that name browsing is implemented in the opc server.

Ge

An item value can be displayed in a Ge graph by using the name in the extern volume. For example, if the mount object for the extern volume is 'Ext-P1', and the local name of the item is

```
/P1/Signals/Ai22
```

the signal name in Ge will probably be (this is dependent of the browsing function of the server)

```
Ext-P1-P1-Signals-Ai22.Value##Float32
```

presuming that it is a float datatype.

Plc

Item values can also be handled in the plc program, using the GetExt... and CStoExt... objects. The normally used objects for getting and storing attributes GetDp, GetAp, StoDp, StoAp etc. can not be used, as they require that the referenced name is known in the development environment, which is not the case for mosts extern volumes. In the Ext objects, the reference is made with a name string, making it possible to enter the item name. To get the value of the item in the previous example, you should use a GetExtFloat32 object, and the object name should be

```
Ext-P1-P1-Signals-Ai22.Value
```

To store a value in an item, lets say /P1/Signals/Ao5, you use a CstoExtFoat32. This objects makes a conditional storage, and only on a positive edge of the condition. Compare with the CstoAp, where the value is stored, as long as the condition is true. The reference name in the CstoExtFloat32 object in this case should be

```
Ext-P1-P1-Signals-Ao5.Value
```

Client process

For each opc client-server connection a client process has to be started. The executable for this process i opc_provider that has the arguments

1. Opc server URL.
2. Extern volume id.
3. Extern volume name.
4. Server identity (optional, default 200).

Configuration

Register ExternVolume

Register the externvolume in the GlobalVolumeList with a volume name and identity.

Application file

Add a line in the application file to start the opc_provider. Here is an example for a opc client connting to the opc server 'http://servernode:8080'. The registered externvolume has the name MyOpcVolume with volume id 0.1.99.55

```
opc_provider, opc_provider, noload, run, opc_provider, 9, nodebug,
http://servernode:8080 0.1.99.55 MyOpcVolume
```

If item values are fetched into the plc, the priority should be set to 4 (sixth argument).

Mount object

Create a mount object in the plant hierarchy of the rootvolume. and insert the objid of the volumeobject in the externvolume into the Object attribute. In the example above this objid is `_O0.1.99.55:0`.

Hint

The application file reside on $pwrp_load and has the name $pwrp_load/ld_appl_'nodename'_'busnumber'.txt where node name is the name of the node, and busnumber the qcom bus number. If the node is 'mynode' and the busnumber is 507, the filename will be

```
$pwrp_load/ls_appl_mynode_507.txt
```

New functions in Ge

Invisible

Dynamic type Invisible can be connected to analog values. When the value is zero, the object is made invisible.

DigColor

Dynamic type DigColor can be connected to a bit in a bitmask. The type specification of the

attribute should be `##Bit['bitnumber']`, where bitnumber for the first bit is 0.

New ClassVolume

Opc

The Opc class volume contains classes for the opc client and server.

New Types

Int64

A base type for a 64 bit signed integer.

UInt64

A base type for a 64 bit unsigned integer.

New Classes

Opc:Opc_ServerConfig

Configuration object for an opc server.

Opc:Opc_ServerState

An object created as first object in the extern volume in an opc client, displaying information about the connected opc server.

Opc:Opc_Hier

An object in the extern volume in an opc client representing a branch.

Opc:Opc_String

An object in the extern volume in an opc client representing an item of datatype string.

Opc:Opc_Boolean

An object in the extern volume in an opc client representing an item of datatype boolean.

Opc:Opc_Float

An object in the extern volume in an opc client representing an item of datatype float.

Opc:Opc_Double

An object in the extern volume in an opc client representing an item of datatype double.

Opc:Opc_Decimal

An object in the extern volume in an opc client representing an item of datatype decimal.

Opc:Opc_Long

An object in the extern volume in an opc client representing an item of datatype long.

Opc:Opc_Int

An object in the extern volume in an opc client representing an item of datatype int.

Opc:Opc_Short

An object in the extern volume in an opc client representing an item of datatype short.

Opc:Opc_Byte

An object in the extern volume in an opc client representing an item of datatype byte.

Opc:Opc_UnsignedLong

An object in the extern volume in an opc client representing an item of datatype unsignedLong.

Opc:Opc_UnsignedInt

An object in the extern volume in an opc client representing an item of datatype unsignedInt.

Opc:Opc_UnsignedShort

An object in the extern volume in an opc client representing an item of datatype unsignedShort.

Opc:Opc_UnsignedByte

An object in the extern volume in an opc client representing an item of datatype unsignedByte.

Opc:Opc_Base64Unary

An object in the extern volume in an opc client representing an item of datatype base64Unary.

Opc:Opc_DateTime

An object in the extern volume in an opc client representing an item of datatype dateTime.

Opc:Opc_Date

An object in the extern volume in an opc client representing an item of datatype date.

Opc:Opc_Duration

An object in the extern volume in an opc client representing an item of datatype duration.

Opc:Opc_QName

An object in the extern volume in an opc client representing an item of datatype QName.

pwrb:GetExtBoolean

A plc object to fetch an attribute value of type Boolean from an extern volume.

pwrb:GetExtFloat32

A plc object to fetch an attribute value of type Float32 from an extern volume.

pwrb:GetExtFloat64

A plc object to fetch an attribute value of type Float64 from an extern volume.

pwrb:GetExtInt8

A plc object to fetch an attribute value of type Int8 from an extern volume.

pwrb:GetExtInt16

A plc object to fetch an attribute value of type Int16 from an extern volume.

pwrb:GetExtInt32

A plc object to fetch an attribute value of type Int32 from an extern volume.

pwrb:GetExtInt64

A plc object to fetch an attribute value of type Int64 from an extern volume.

pwrb:GetExtUInt8

A plc object to fetch an attribute value of type UInt8 from an extern volume.

pwrb:GetExtUInt16

A plc object to fetch an attribute value of type UInt16 from an extern volume.

pwrb:GetExtUInt32

A plc object to fetch an attribute value of type UInt32 from an extern volume.

pwrb:GetExtUInt64

A plc object to fetch an attribute value of type UInt64 from an extern volume.

pwrb:GetExtString

A plc object to fetch an attribute value of type String from an extern volume.

pwrb:GetExtTime

A plc object to fetch an attribute value of type Time from an extern volume.

pwrb:CStoExtBoolean

A plc object for conditinal storage of a Boolean value into an attribute in an extern volume.

pwrb:CStoExtFloat32

A plc object for conditinal storage of a Float32 value into an attribute in an extern volume.

pwrb:CStoExtFloat64

A plc object for conditinal storage of a Float64 value into an attribute in an extern volume.

pwrb:CStoExtInt8

A plc object for conditinal storage of a Int8 value into an attribute in an extern volume.

pwrb:CStoExtInt16

A plc object for conditinal storage of a Int16 value into an attribute in an extern volume.

pwrb:CStoExtInt32

A plc object for conditinal storage of a Int32 value into an attribute in an extern volume.

pwrb:CStoExtInt64

A plc object for conditinal storage of a Int64 value into an attribute in an extern volume.

pwrb:CStoExtUInt8

A plc object for conditinal storage of a UInt8 value into an attribute in an extern volume.

pwrb:CStoExtUInt16

A plc object for conditinal storage of a UInt16 value into an attribute in an extern volume.

pwrb:CStoExtUInt32

A plc object for conditinal storage of a UInt32 value into an attribute in an extern volume.

pwrb:CStoExtUInt64

A plc object for conditinal storage of a UInt64 value into an attribute in an extern volume.

pwrb:CStoExtString

A plc object for conditinal storage of a String value into an attribute in an extern volume.

pwrb:CStoExtTime

A plc object for conditinal storage of a Time value into an attribute in an extern volume.

pwrb:AtoFloat64

A plc object to convert a Float32 value to Float64.

pwrb:Float64toA

A plc object to convert a Float64 value to Float32.

pwrb:Int64toI

A plc object to convert a Int64 value to Int32.

pwrb:ItoInt64

A plc object to convert a Int32 value to Int64.

pwrb:UInt64toI

A plc object to convert a UInt64 value to Int32.

pwrb:ItoUInt64

A plc object to convert a Int32 value to UInt64.

pwrb:UInt32toI

A plc object to convert a UInt32 value to Int32.

pwrb:ItoUInt32

A plc object to convert a Int32 value to Uint32.

V4.4.1 additions

Status Server

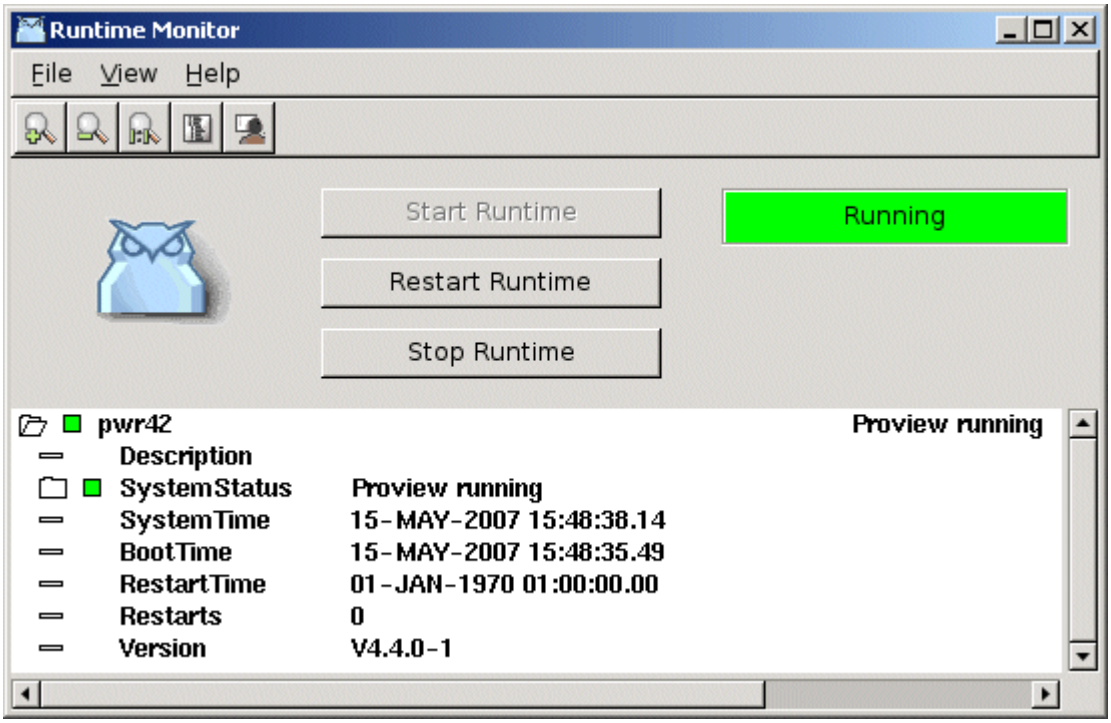
The Status Server is a server process that implements a web service that responds to a number of request to get status information about a process or operator station, and to start xtt on the current display. The Status Server has to be started if the Runtime Monitor or the Supervision Center is used (see below).

The Status Server (rt_statussrv) is started if a StatusServerConfig object is placed in the node hierarchy under the $Node object.

Runtime Monitor

The runtime monitor starts, stops and monitors the runtime environment started on a development

station. It is opened from the menu in the navigator T*ools/Runtime Montor.*



The runtime environment is started with the pushbutton *Start Runtime* and stopped with the pushbutton *Stop Runtime*. A soft restart is executed by the pushbutton *Restart Runtime*. *File/Reset* in the menu removes the runtime processes and shared memory, this sometimes has to be done, when the runtime environment is not correctly started or terminated. The current runtime status is displayed by the colored rectangle to the left, and in the lower part of the window. Xtt with or without operator window can be started from the menu or from buttons in the tools panel.

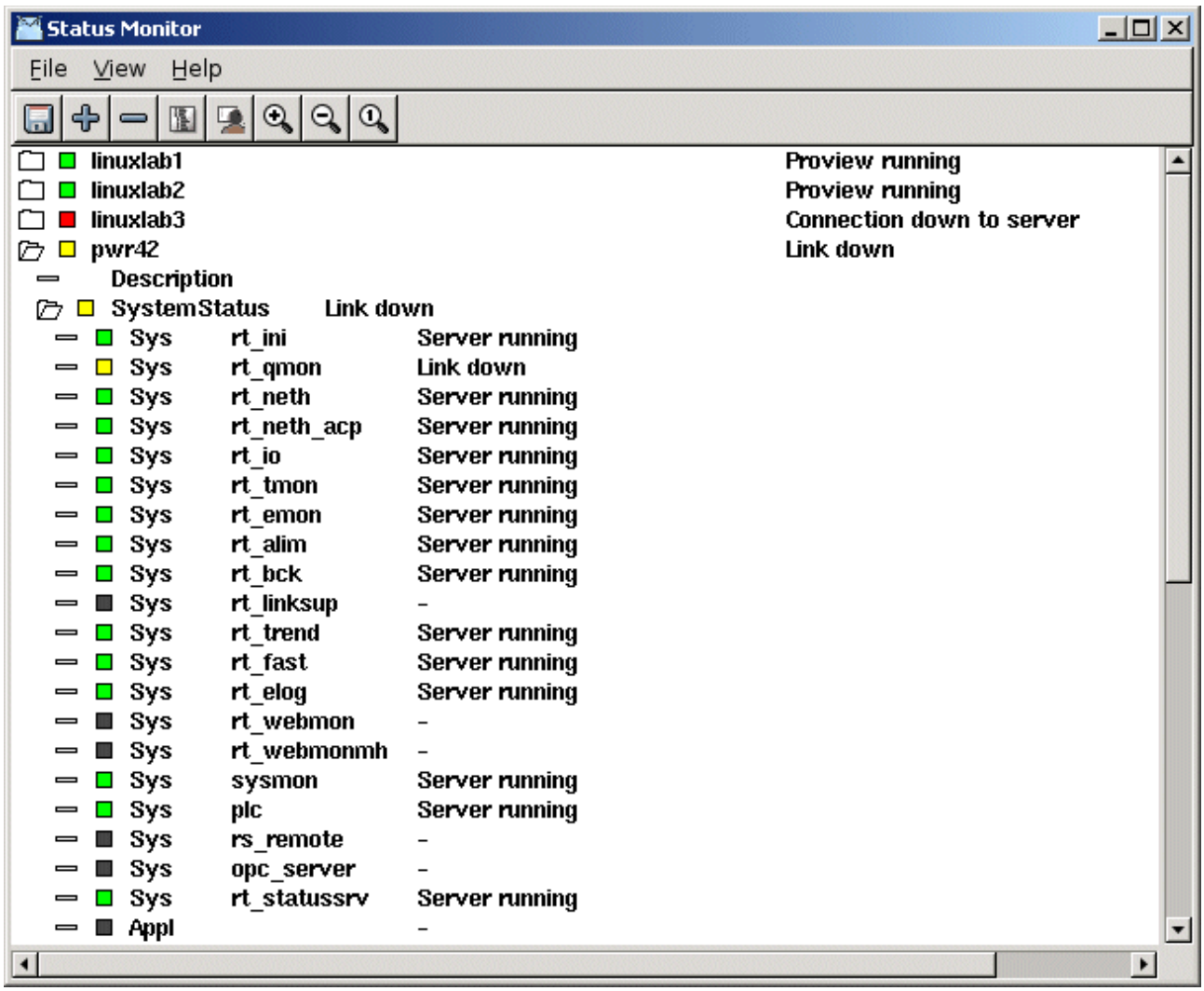
Supervision Center

From the Supervision Center, a number of process and operator stations can be supervised. The system status for the station are fetched via the status monitor, that has to be started in the supervised nodes. New nodes are added by *File/Add Node* or the + button in the tools panel. For each node, the Description of the $Node object, the SystemStatus, SystemTime, BootTime, RestartTime, number of restarts and proview version are displayed. The SystemStatus can be opened, and then, the status of each system process add application process are displayed.

It is also possible to start xtt, with or without an operator window from the menu (F*ile/Open Runtime Navigator* and *File/Open Operatorplace*). To open an operator place, a valid OpPlace object has to be supplied when the node is added.

All changes of system status are stored in an eventlist, and by activating *View/Pop Events* the eventlist is popped on top of the screen when an error status is received.

The current configuration of process and operator stations are stored with *File/Save Configuration*.



New classes

pwrb:StatusServerConfig

Object to start the Status Server. Should be placed in the node hierarchy under the $Node object.

Upgrade procedure

The upgrading has to be done from V4.3.0. If the project has a lower version, the upgrade has to be performed stepwise following the schema

V2.1 -> V2.7b -> V3.3 -> V3.4b -> V4.0.0 -> V4.1.3 -> V4.2.0 ->V4.4.0/V4.4.1

The upgrade procedure is to change the version of the project in the projectlist, and then activate Functions/Update Classes from the meny in the Configurator. If the V4.3 version should be kept, first make a copy of the project.

Make a copy of the project

Do `sdf` to the project and start the administrator

```
> pwra
```

Now the Projectlist is opened. Enter edit mode, login as administrator if you lack access. Fined the current project and select Copy Project from the popup menu of the ProjectReg object. Open the copy and assign a suitable project name and path. Change the version to V4.4.0/V4.4.1. Save and close the administrator.

Update Classes

Do `sdf` to the project.

Start the Configurator, enter edit mode, and activate Functions/Update Classes in the menu.

Save and Build.