



ProviewR  
OPEN SOURCE PROCESS CONTROL

# Release Notes V5.6

2018 11 06

Copyright © 2005-2018 SSAB EMEA AB

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

# Table of Contents

Upgrading to ProviewR V5.6.0.....	5
New functions.....	5
Revisions.....	5
Revision window.....	5
Create a revision.....	5
Restore a revision.....	6
Create a branch.....	6
New files.....	6
Configurator alternatives for object and type attributes.....	7
CompPosit.....	7
Remote RabbitMQ.....	8
Thread safe strings and times.....	9
Time lock.....	9
String lock.....	10
IO copying of ATv, DTv and Sv.....	10
Ge object navigator.....	10
Navigator filter.....	11
Ge syntax control.....	11
Xtt curve window features.....	13
Curve type.....	13
Line and points.....	13
Points.....	14
Square.....	14
Split digital curves.....	15
Storage station features.....	16
Realtime database.....	16
Configuration.....	16
Item object tree.....	19
SevPlotGroup.....	20
Mysql threads.....	20
New sev parameters in /etc/proview.cnf.....	21
sevMeanValue1All.....	21
sevLinearRegrAll.....	21
sevLinearRegrMaxTime.....	22
Ge Window/TabbedWindow object graph depth.....	22
Ge display of parent attributes in object graphs.....	22
Variable array index in Ge graphs.....	22
Ge AnalogColor instances with separate attributes.....	22
Crossreferences options in configurator settings.....	22
Xtt commands to set signal invert, conversion and test.....	23
Xtt menu methods to set signal, invert and test.....	23
Show all toplevel objects in the configurator.....	23
Display of Profinet configuration in Xtt.....	24
Profinet viewer displays the configured devices.....	24
Plotgroup objects in OpPlace.FastAvail.....	24
Conversion functions cdh_XxxToString() modified.....	24
Web interface in HTML5 and javascript.....	25
Configuration.....	25
New Types and Classes.....	26
CurveLayoutMask.....	26

TimeRangeEnum.....	26
SevPlotGroup.....	26
SevItem, SevItemInteger, SevItemFloat and SevItemBoolean.....	26
DataArithmT and DataArithmTL.....	26
CompPosit.....	27
WebSocketServer.....	27
Modified Classes.....	27
PlotGroup.....	27
Di.....	27
Do.....	27
Upgrade procedure.....	27
Make a copy of the project.....	27
Dump the databases.....	28
Linux release upgrade.....	29
Change version.....	29
upgrade.sh.....	29
savedirectory.....	29
classvolumes.....	29
cnvdump.....	29
renamedb.....	29
loaddb.....	29
cnvobjects.....	29
compile.....	29
createload.....	30
createboot.....	30
List example.....	30

# Upgrading to ProviewR V5.6.0

This document describes new functions in ProviewR V5.5.0, and how to upgrade a project from V5.5.0 to V5.6.0.

## New functions

### Revisions

A revision is a state in the development environment that is stored in a version control system. The revision should include all source files to make it possible to restore a revision. A revision can be used to review the state when the revision was created, and also to build runnable systems from the restored source to test or run a restore revision in production.

The currently supported version control system is Git. A new repository will be created on \$pwrp\_root/src when the first revision is created.

### Revision window

Revisions are handled from the Revision window that is opened from the File menu in the configurator.

### Create a revision

A new revision is created from Functions/New Revision in the menu. New revisions can only be created if the currently checked out revision is the end point of a branch.

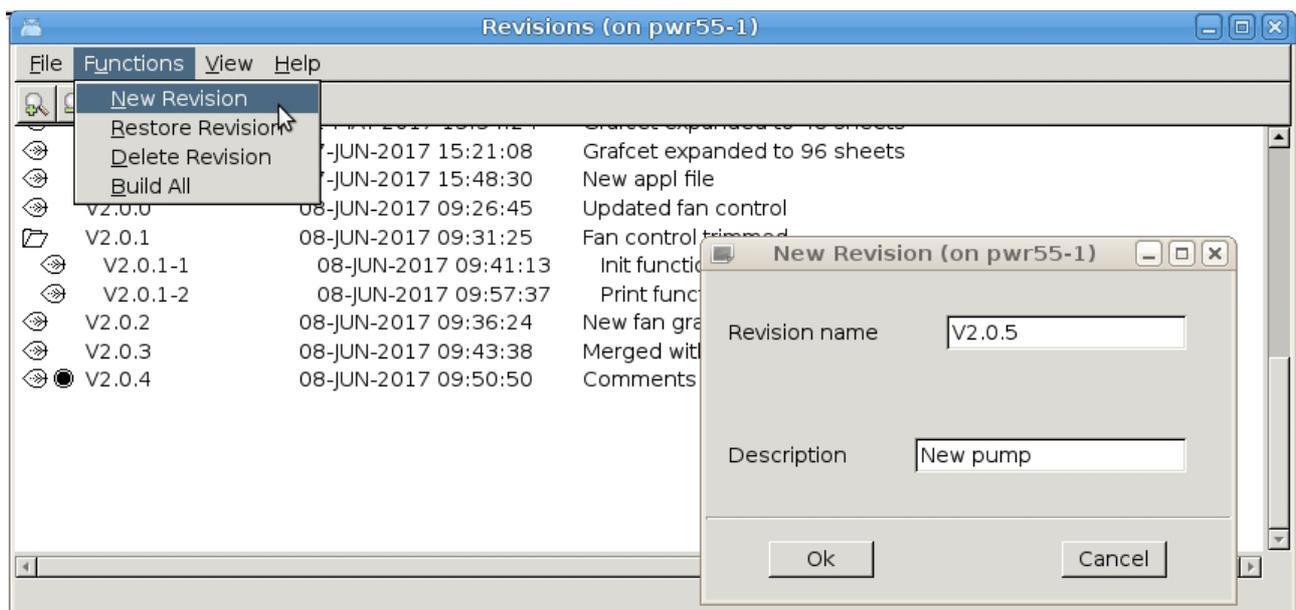
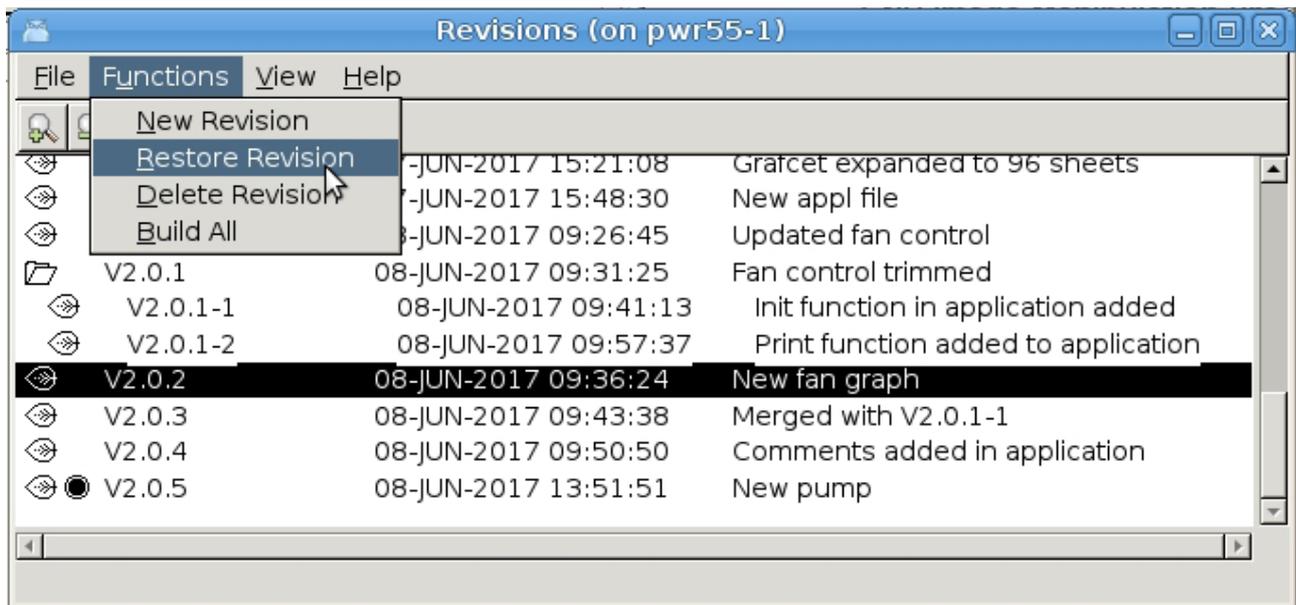


Fig Create a new revision

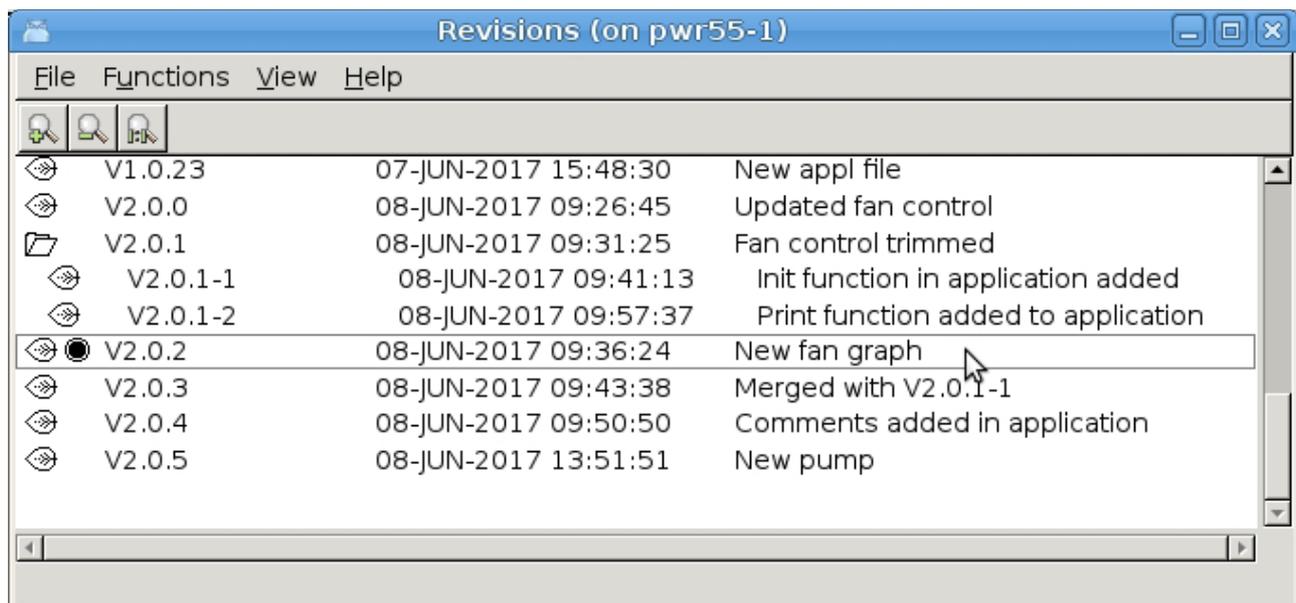
## Restore a revision

A revision is restored from Functions/Restore in the menu. Select the revision to restore first.



**Fig Restore a revision**

When a revision is restored the checked out revision is marked with a radio button in the list. Also in the configurator title bar, the current revision is written.



**Fig An old revision is restored**

## Create a branch

If an old revision in the main line is restored, a branch can be created by creating a new revision from this point. In the figure above, a branch has been created from revision V2.0.1. New revisions can also be created from the end point of the branch. In the figure above, a new revision can be created from V2.0.1-2 that is an end point, but not from V2.0.1-1.

## New files

New files, e.g. c-applications, are not automatically added to the git repository. This has to be done manually. New files are displayed by Git as untracked files in the 'git status' command. Untracked

files can be added with the 'git add' command and will then be included in the next revision.

Note! It's important to add new files to be able to restore a complete revision.

### Example

```
> git status
On branch B_V2.0.5
Untracked files:
(use "git add <file>..." to include in what will be committed)
appl/ra_appl.c
> git add appl/ra_appl.c
```

The files that automatically are added to the repository are wb\_rev-files on \$pwrp\_db and pwg-files in XttGraph, WebGraph and AppGraph objects. All other files has to be inserted manually.

## Configurator alternatives for object and type attributes

When inserting values for attributes of type object identity and attribute references, you previously had to type the object name, or find it in the hierarchy somewhere and connect or copy the name. Now the valid objects of the attribute will be listed when the attribute is opened. In the example below, the ThreadObject attribute in the PlcPgm is opened, and the present PlcTreads objects are displayed as alternatives.

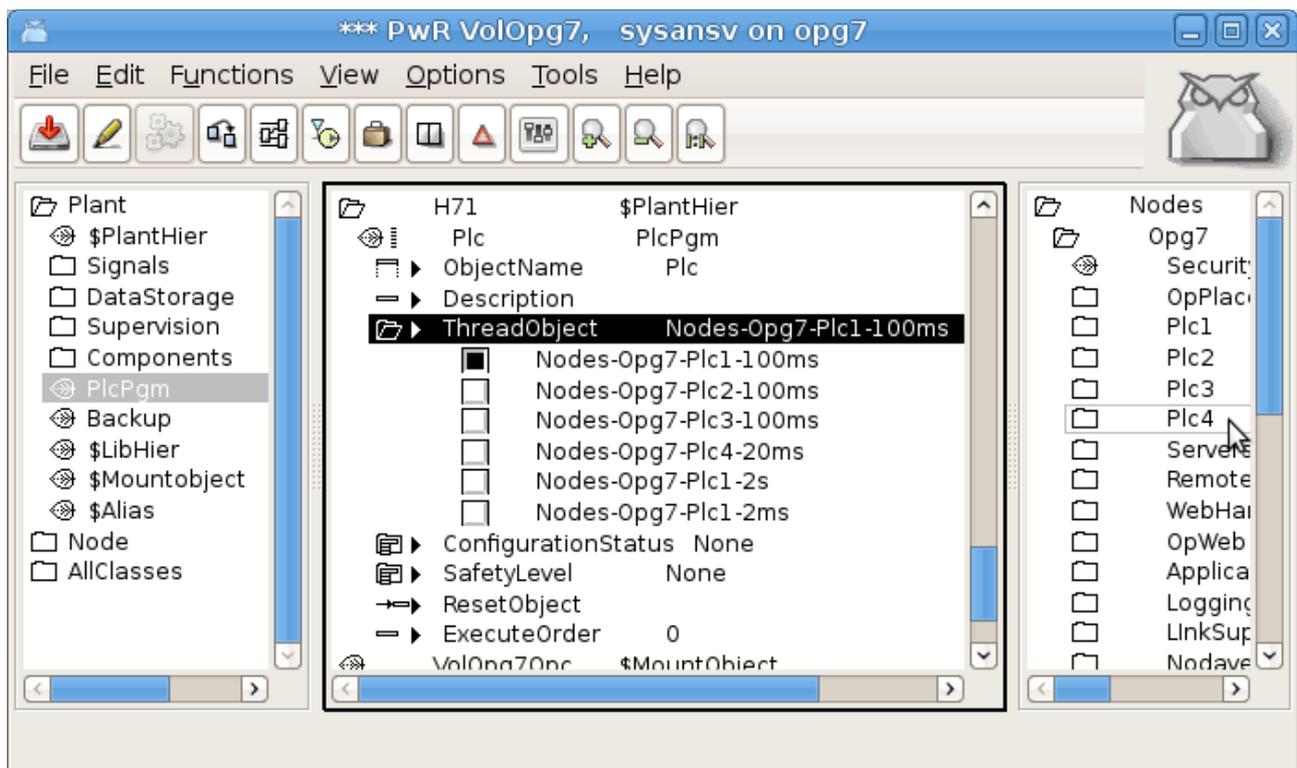


Fig The alternatives for the PlcPgm.ThreadObject

The alternatives are configured in the class description with a \$ReferenceList object. The \$ReferenceList object is positioned under the ClassDef object and named after the attribute it serves.

## CompPosit

CompPosit is a positioner implemented as a component with main object and function object.

The positioner calculates an output from the current position and a desired set position. When the distance to the set position is larger than a break distance, the maximum output is set as output. When the distance is less than the break distance the output is proportional to the square root of the distance. When the distance enters the dead zone, the unit is considered to be in position. The positioning continues beyond that for a period of time, InPosTime, before it is stopped. It can also be stopped from the program (in Auto mode) or by the operator (in Manual mode).

The operating mode, Auto or Manual, is selected in the object graph.

In Auto mode, the object is controlled by inputs in the function object.

The set position is fetched from the SetPosValue input, and the positioning is started on leading edge of the Start input. It's stopped from the Reset input, or automatically when the desired position is reached if TimerStopPosit is set.

In Manual mode, the object is controlled from the object graph. The set position can be entered in the Set field, and start/stop is controlled from the Start and Stop buttons.



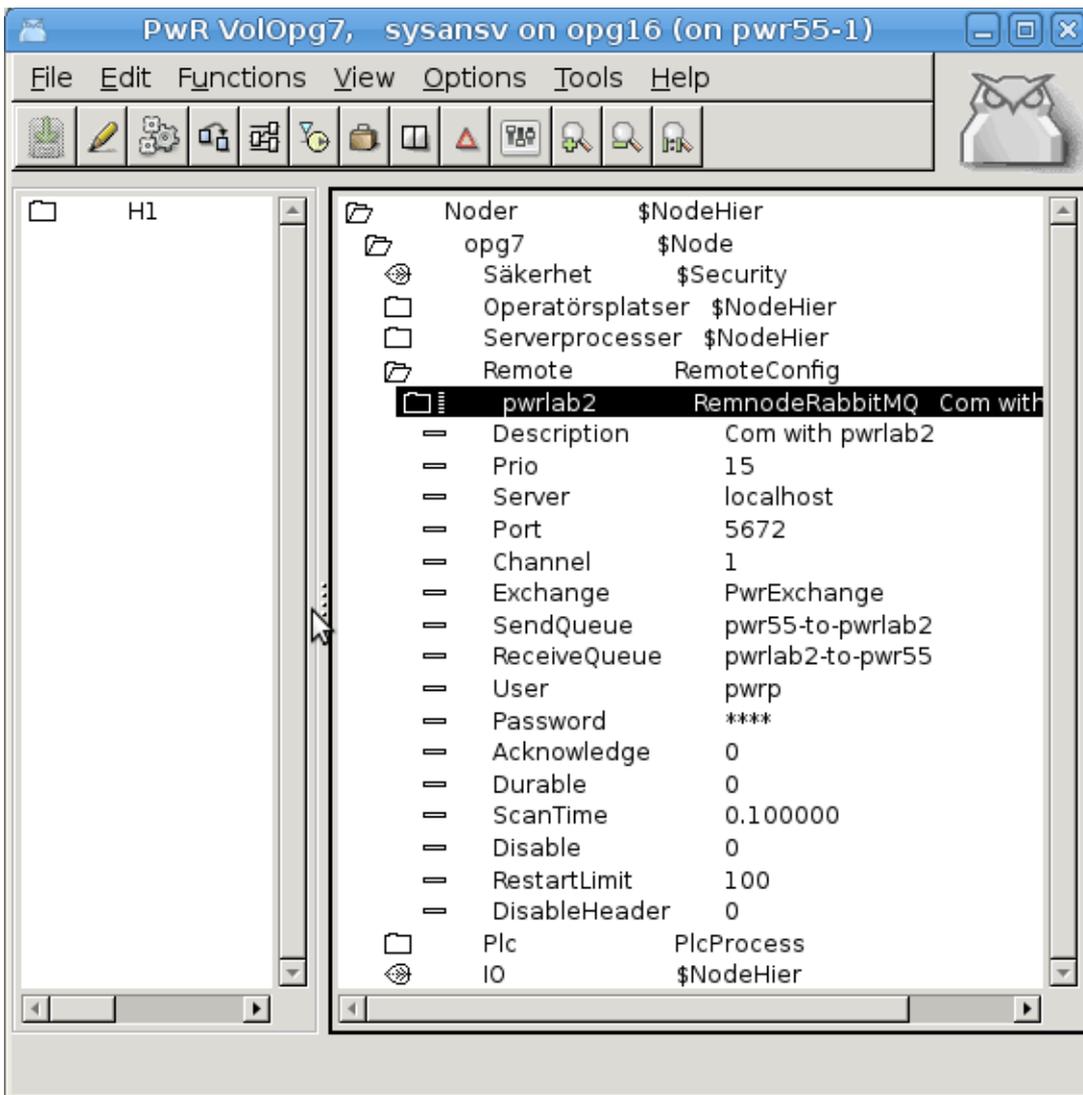
Fig Object graph for CompPosit

## Remote RabbitMQ

Communication with the message queue RabbitMQ is implemented in the remote module. Configuration is made with a RemnodeRabbitMQ object where the RabbitMQ server is specified. Also a send queue and a receive queue can be specified. The queues will be created on the server if they don't already exist. Messages can be sent with acknowledge which means that messages will remain in the queue until they are acknowledged by the receiver. Messages can also be store on disk with the Durable attribute. In this case both the queue and the messages will be restored after a server breakdown.

The server require login with username and password. Users are created on the server with rabbitmqctl.

Read more about the remote function in Designer's Guide.



**Fig Remote node configuration of RabbitMQ**

## ***Thread safe strings and times***

Times and strings can not be accessed in an atomic operation and has thus not been thread safe. Therefore two new locks are introduced, one for times and one for strings.

### **Time lock**

The time lock is used for attributes of type `pwr_tTime` and `pwr_tDeltaTime`. Every time a time attribute is read or written, the lock should be set. The following function objects will use the lock to protect the reading or writing of the time value.

`StoATv`, `StoDTv`, `CStoATv`, `CStoDTv`, `StoATp`, `StoATp`, `CStoATp`, `GetATp`, `GetATv`, `GetDTp`, `GetDTv`, `CurrentTime`.

Note that other time objects are not protected by the lock. It's not safe to use the output from for example a `AtAdd` in another Plc thread or an application program. In this case the time should first be stored in an `ATv` object.

Applications should use the new `gdh` functions

```
void gdh_GetTimeDL( pwr_tTime *atp, pwr_tTime *time);
void gdh_SetTimeDL( pwr_tTime *atp, pwr_tTime *time);
void gdh_GetDeltaTimeDL( pwr_tDeltaTime *dtp, pwr_tDeltaTime *time);
```

```
void gdh_SetDeltaTimeDL( pwr_tDeltaTime *dtp, pwr_tDeltaTime *time);
pwr_tStatus gdh_GetObjectInfoTime( char *name, pwr_tTime *time);
pwr_tStatus gdh_SetObjectInfoTime( char *name, pwr_tTime *time);
pwr_tStatus gdh_GetObjectInfoDeltaTime( char *name, pwr_tDeltaTime *time);
pwr_tStatus gdh_SetObjectInfoDeltaTime( char *name, pwr_tDeltaTime *time);
```

Before using any of these functions, the time lock has to be initialized by the application with a call to `lck_Create()`, eg

```
lck_Create(&sts, lck_eLock_Time);
```

## String lock

The string lock is used for string attributes. The following objects will use the lock.

StoSv, CStoSv, StoSp, CStoSp, StoNumSp, CStoNumSp, GetSp, GetSv.

Other string objects are not protected by the lock and the string values in these objects should not be read or written in other Plc threads or in application programs.

Applications should use the new `gdh` functions

```
void gdh_GetStrDL( char *sp, char *str, int size);
void gdh_SetStrDL( char *sp, char *str, int size);
pwr_tStatus gdh_GetObjectInfoStr( char *name, char *str, int size);
pwr_tStatus gdh_SetObjectInfoStr( char *name, char *str, int size);
```

Before using any of these functions, the string lock has to be initialized by the application with a call to `lck_Create()`, eg

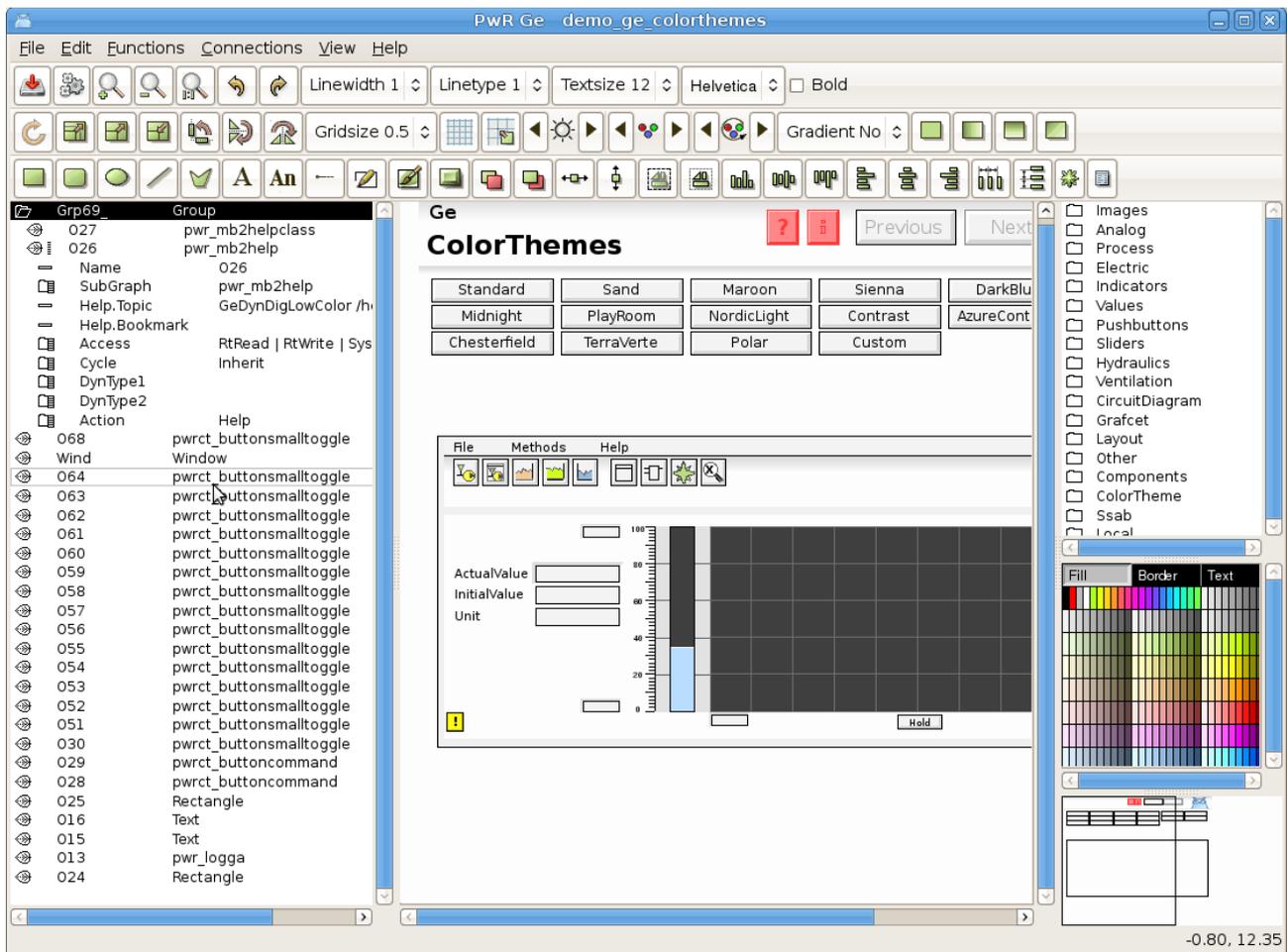
```
lck_Create(&sts, lck_eLock_Str);
```

## IO copying of ATv, DTv and Sv

As most other signals, ATv, DTv and Sv objects are now IO copied. This means that ActualValue now a pointer, pointing to an area object where the actual signal value is stored. An initial value has to be set in the InitialValue attribute, not in ActualValue as before.

## *Ge object navigator*

A new navigator that displays the Ge objects in a tree structure is implemented.



**Fig Ge object tree**

The object tree is displayed by activating 'View/View Object Tree' in the menu. It displays all objects and groups in the graph. An object can be opened displaying the properties of the object. A group is displayed as a map, and group members are displayed as children to the group. Properties for the members can also be opened and modified. Objects can be moved up or down which will affect if objects are drawn on top of below each other. When an object is selected in the tree, it's also selected in the drawing area which makes it easy to identify the objects.

## Navigator filter

The object tree can be filtered to only display objects that are interesting for the moment. The filtering is done with the 'filter navigator' command. For example 'filter navigator /type=name /pattern=\*button\*' will show all objects with the string 'button' in the name. The syntax for the filter function is

```
ge> filter navigator /type= /pattern=
ge> filter navigator /reset
```

```
/reset <t>Reset filter and show all objects.
/type <t>'name' or 'class'. Name will filter on object name
      <t>and class on subgraph.
/pattern <t>Filter pattern. Can contain wildcard (*).
```

## Ge syntax control

The Ge syntax is activated from 'File/Syntax Check' in the Ge menu. It will go through all objects in the graph and check for example that all mandatory properties are filled in, that referenced attributes

exist and are of the correct type and that format specifications are correct. Detected errors and warnings are displayed in the Message window, and by clicking on the arrow the corresponding object is selected in the graph and in the object tree.

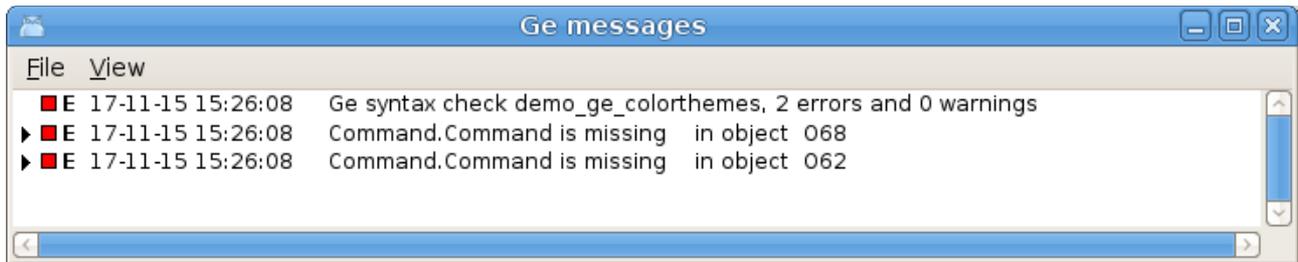


Fig Ge syntax error messages

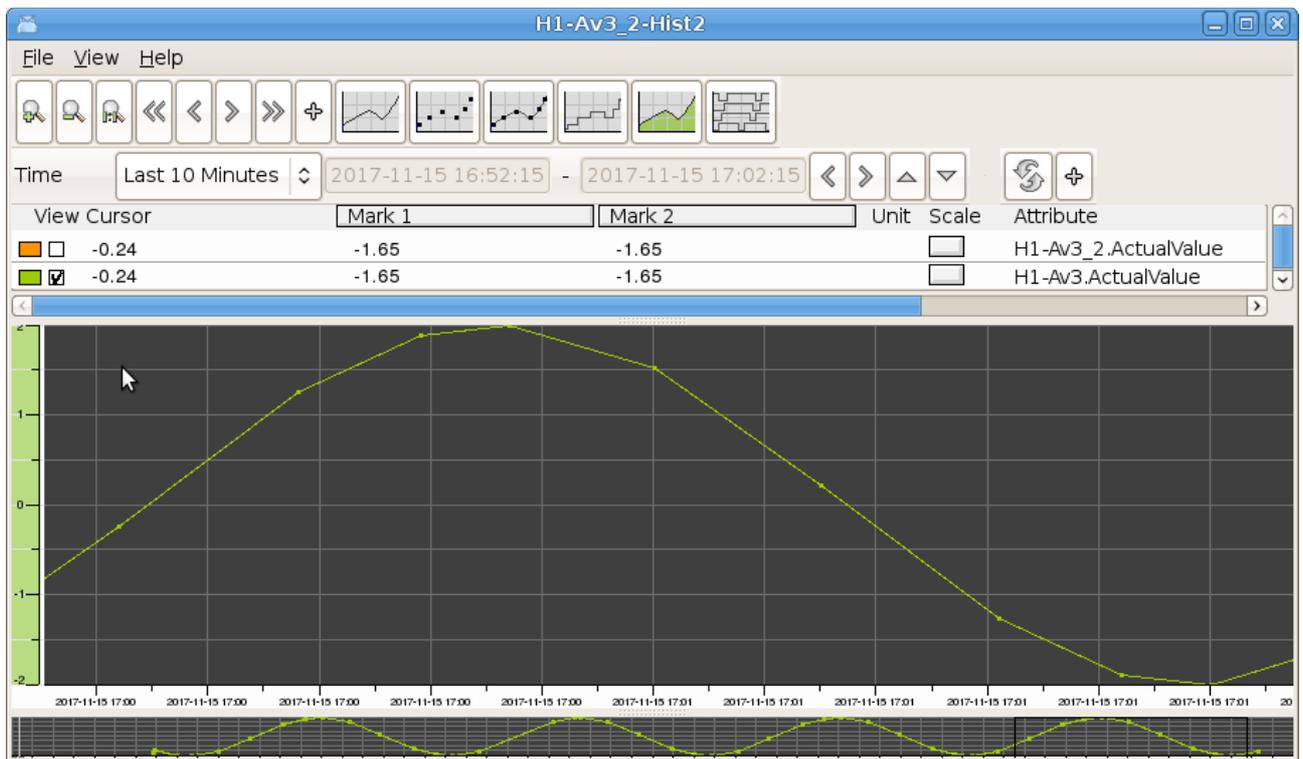
## ***Xtt curve window features***

### **Curve type**

The curve type can be selected with four buttons in the toolbar. In the original curve type the curve is drawn with straight lines between the points. In addition to this the curve types *Line and points*, *Points* and *Square* can be selected.

### ***Line and points***

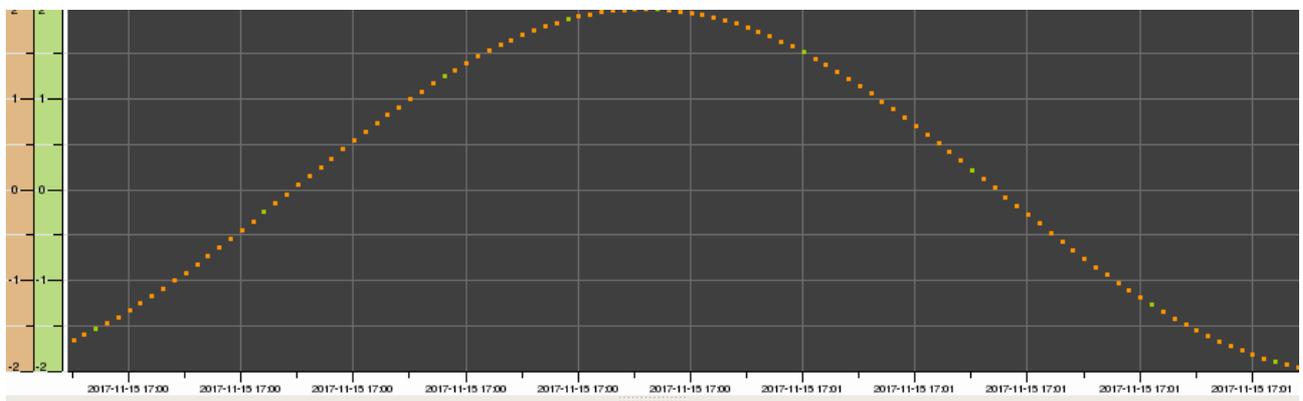
This curve type will mark the stored points and draw straight lines between them.



**Fig Curve type Line and points**

### **Points**

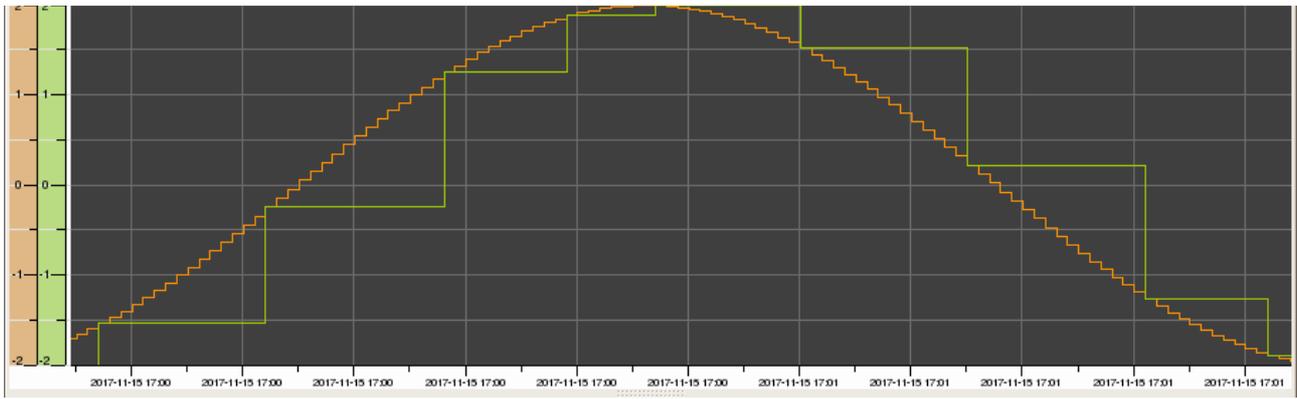
This curve type will draw the points but no lines.



**Fig Curve type Points**

### **Square**

Curve type with horizontal lines between the stored points.



## Split digital curves

Digital values are drawn with square lines, and is by default drawn on the bottom line. With the 'Split digital curves' button in the toolbar, they can be separated and spread vertically over the curve window.

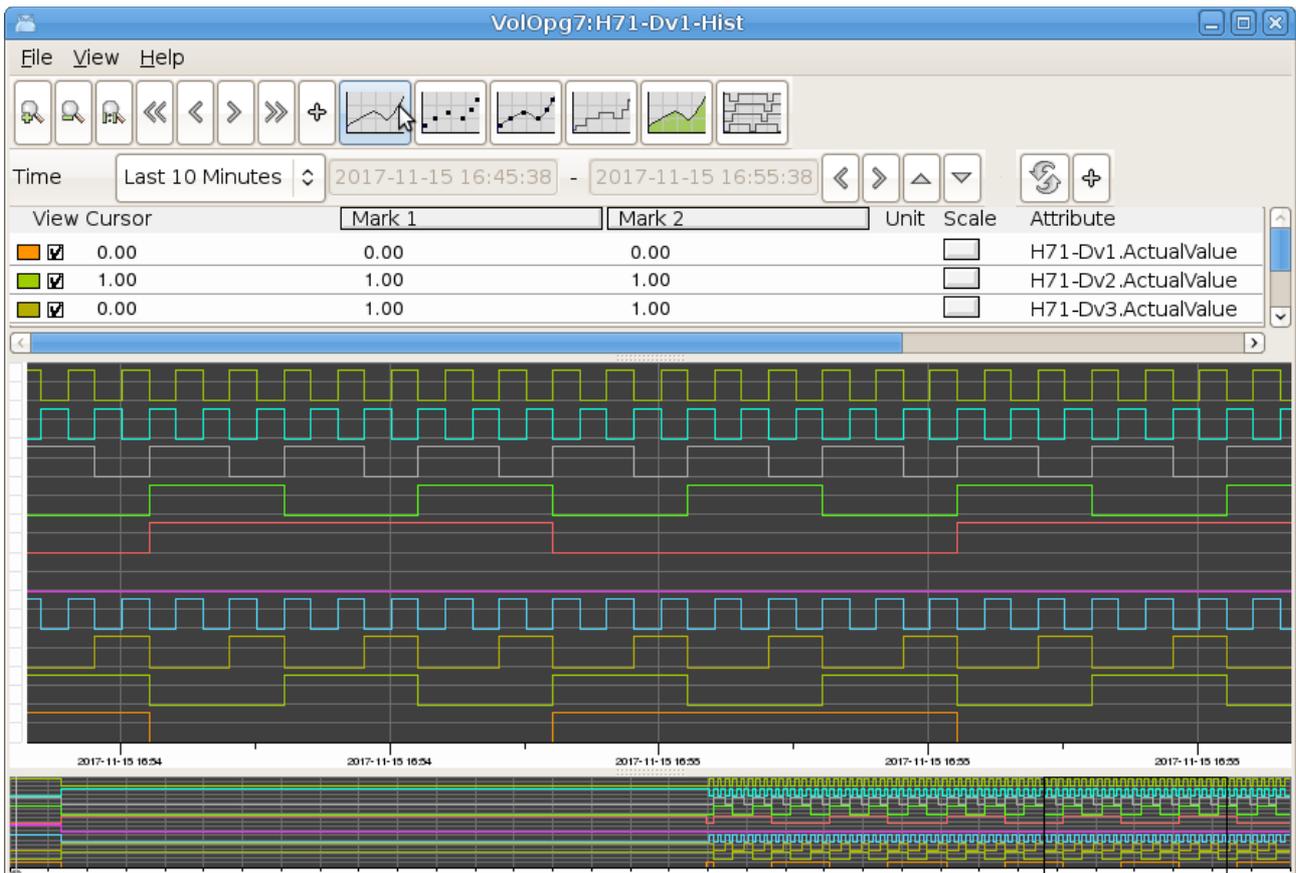


Fig Split digital curves

## Storage station features

### Realtime database

It's now possible to configure and start the realtime database on a storage station. The realtime database will contain information about the store items (see *Item object tree* below) and make it possible for applications to get information about table names, mean values etc. It is also possible to configure an operator place configured with the new SevPlotGroup objects.



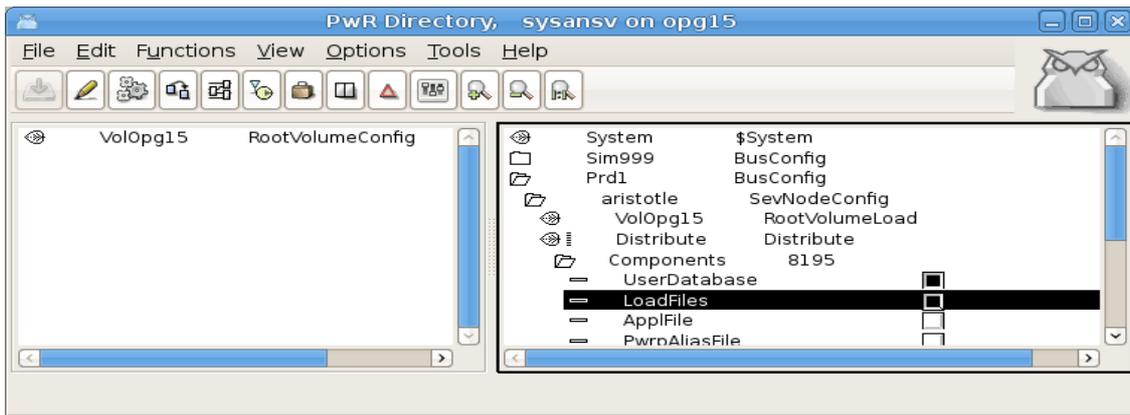
Fig Operator place on a storage station

### Configuration

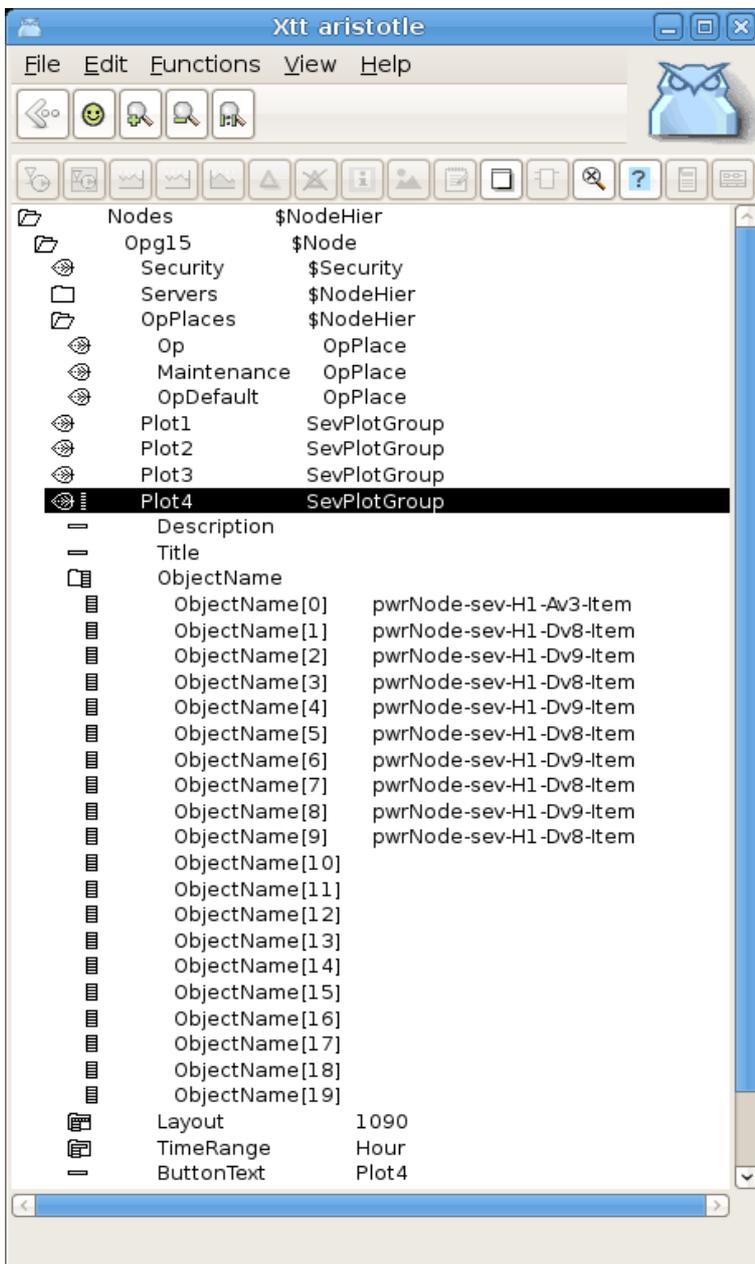
As before a root volume has to be registered for the storage station, as the volume identity for the volume works as the network address for QCom. The new thing is that the root volume can be configured with a RootVolumeConfig object in the Directory volume and opened in the configurator. The configuration should contain a SevServer object in the node hierarchy, and it's also possible to create OpPlace objects and plot group objects to display several curves together in one window on the storage station.

The root volume should be built from the configurator with *Functions/Build/Build Volume* in the menu.

To distribute the root volume load files to the storage station, *LoadFiles* should be marked in the Components attribute of the Distribute object under the SevNodeConfig object in the directory volume.



**Fig LoadFiles is marked in the Distribute object**



**Fig Node hierarchy in a storage station**

## Item object tree

The sev\_server will create an *Item object tree* in the realtime database. The tree is created in the System volume that is mounted under pwrNode. Under pwrNode-sev a tree with the ordinary hierarchy names for the stored signals are built, and as end nodes objects of type SevItemFloat, SevItemInt and SevItemBoolean are created. The SevItem object contains configuration data for the item and also data to monitor the storage of the item. For example the number of received values is displayed, and number of stored values. WriteQuota is the qouta between these values in percent. Also the last received value and the time is displayed.

A mean value is calculated and the configuration of this might seem a bit complicated, and the reason for this is to be backward compatible. It's possible to configure two mean value intervals in /etc/proview.cnf with the sevMeanValueInterval1 and sevMeanValueInterval2 parameters

```
sevMeanValueInterval1 10  
sevMeanValueInterval2 30
```

For each item one of these two intervals can be selected for the mean value calculation. This is done by setting bit MeanValue1 or MeanValue2 in options in the SevHist object. The calculated mean value and standard deviation are displayed in the SevItem object together with the time for the calculation.

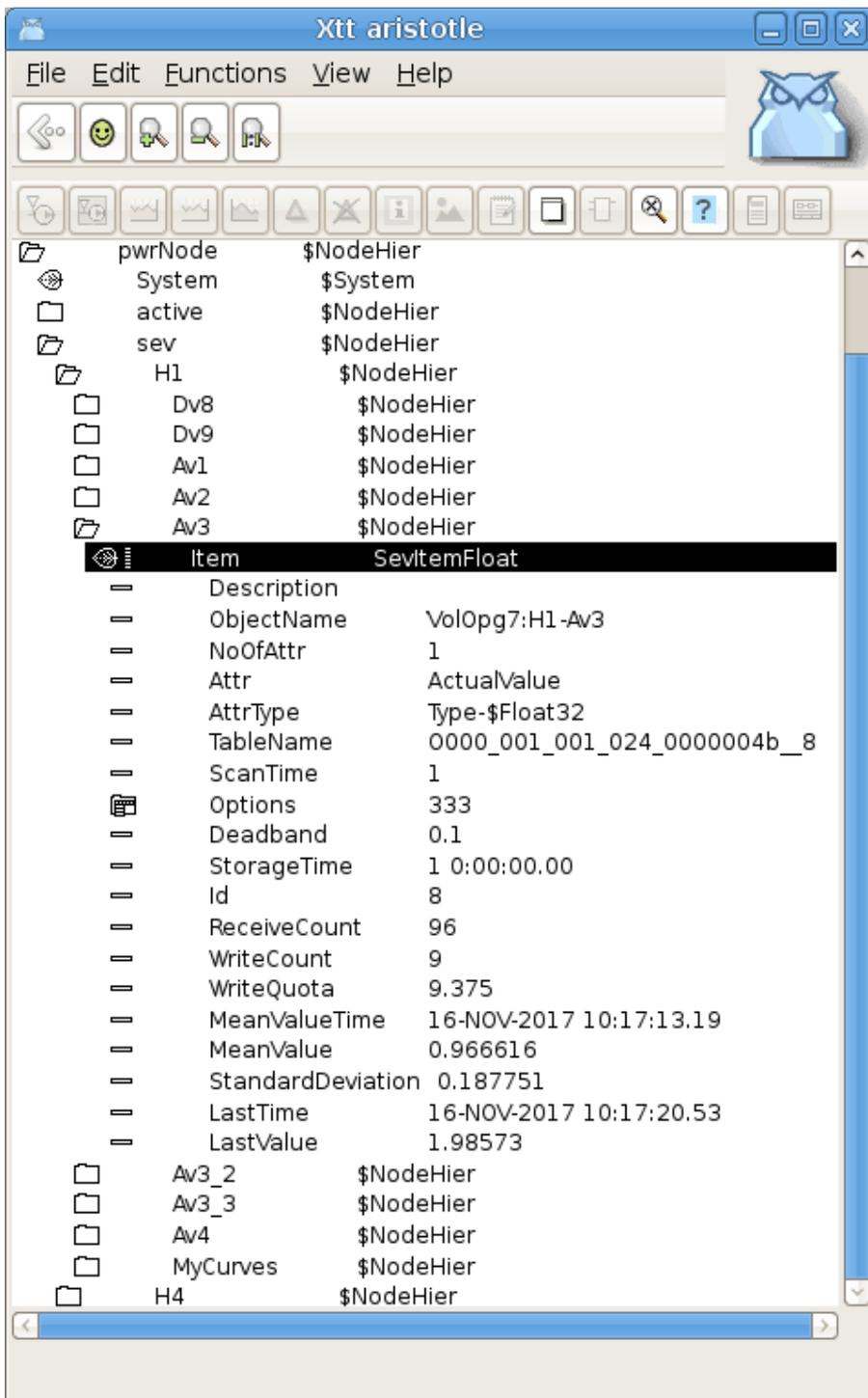


Fig Object item tree

## SevPlotGroup

The SevPlotGroup object makes it possible to configure plot groups in the storage station. An example is viewed in *Fig Node hierarchy in storage station* above. In the ObjectName array SevItem objects in the item object tree is specified, and the curves for all objects in the array will be displayed in the curve window. Note that the names are specified with strings, not object identities. Thus these has to be updated if object names or hierarchies are changed.

## Mysql threads

When using a mysql database it is possible to separate the storage from different messages into

different threads, and thus in different kernels. The sevhist monitor threads can be directed to different storage thread with the new ServerThread attribute in the SevHistThread object. This is integer index value and all values from SevHistThread object with the same index will be handled by the same server thread. In this way the storage can be directed to different storage threads to distribute the load.

It's also possible to sevThreadKeyNode to 1 in /etc/proview.cnf to distribute values from each node to a separate thread. This is useful when communicating with nodes with older version that don't have the ServerThread attribute.

Statistics for the threads are calculated and stored in the SevServer object and is displayed in the object graph for this object.

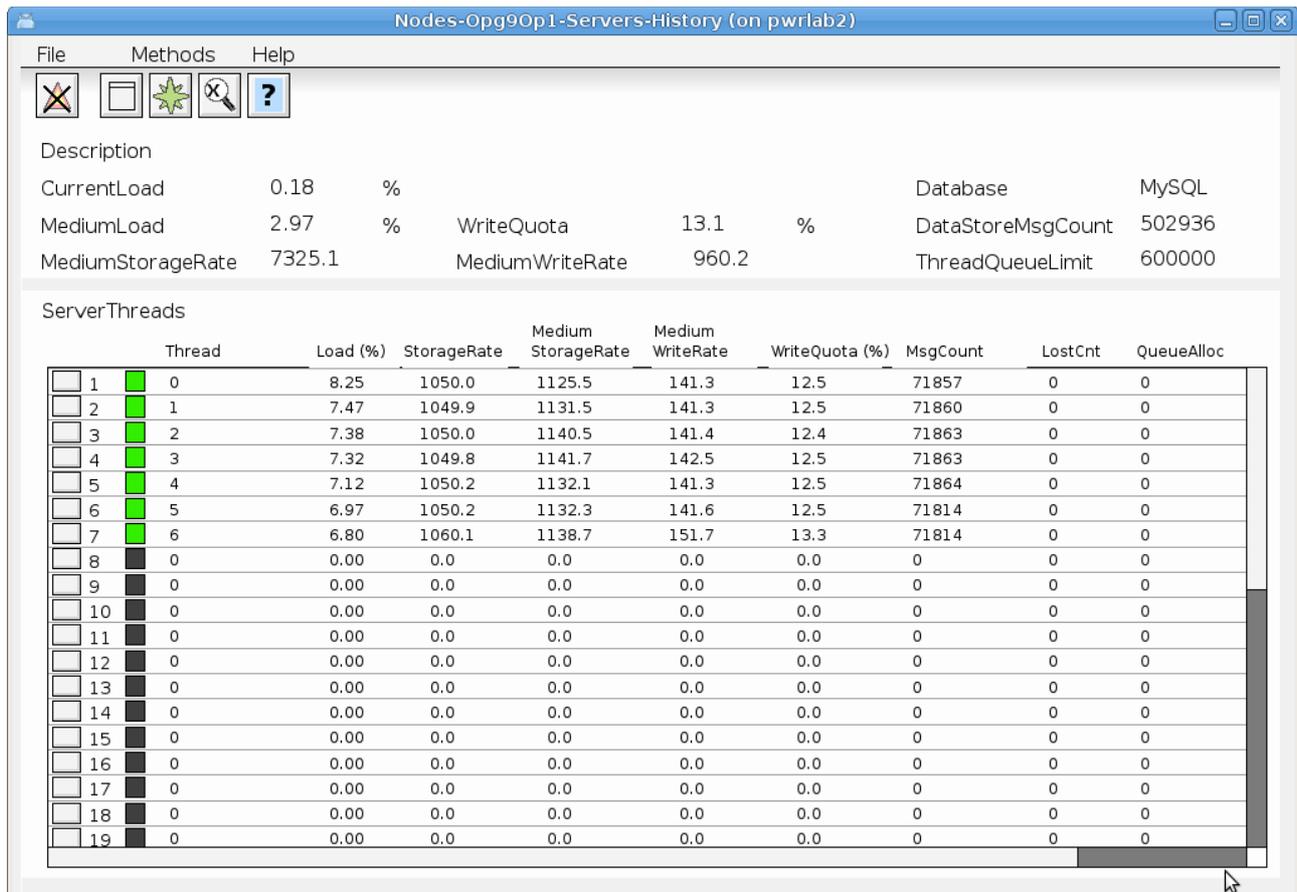


Fig Object graph for SevServer object

## New sev parameters in /etc/proview.cnf

### **sevMeanValue1All**

Parameter to set the MeanValue1 option on all items. This makes it possible to configure mean value calculation also on items from ProviewR nodes with older versions.

```
sevMeanValue1All 1
```

### **sevLinearRegrAll**

Parameter that set the DeadBandLinearRegr option on all items with UseDeadBand. This makes it possible to use the linear regression deadband also on items from ProviewR nodes with older versions.

sevLinearRegrAll 1

### ***sevLinearRegrMaxTime***

Parameter to force a storage within the specified time even if the value is not outside the deadband. Time in seconds.

sevLinearRegrMaxTime 15

### ***Ge Window/TabbedWindow object graph depth***

The depth for object graphs in Windows and TabbedWindows was previously limited to two levels. This limitation is now removed.

### ***Ge display of parent attributes in object graphs***

Attributes in the parent object for an object in an object graph can be referenced with the .< syntax. The parent object to the \$object can be referenced with '.'<' or '-<'. These operators will remove the last attribute or the last object to the closest preceding '.' or '-'. H1-H2-<.Description will be H1.Description. If you have an hierarchy graph for H1-H2, the description in the parent object H1 can be displayed with

```
$object-<.Description##String80
```

For attribute objects, where attribute names is removed instead, '.'<' is used.

H1-Plates.Data.<.Temperature will be H1-Plates.Temperature. Thus in the object graph for Data the Temperature attribute in parent object Plates can be displayed by

```
$object.<.Temperature##Float32
```

### ***Variable array index in Ge graphs***

Array indexes can be specified with attributes of type Int32 or UInt32. In

```
H1-H2-Array[&(H1-H2-CurrentIndex.ActualValue)].Value##Float32
```

H1-H2-CurrentIndex.ActualValue is an Int32 attribute, and the value of this attribute will be fetched and inserted as array index.

The index is evaluated when the graph is opened and if the index value is changed, the connection will not be automatically changed. The dynamic DigSwap though can be used to reconnect all connections in a graph.

### ***Ge AnalogColor instances with separate attributes***

AnalogColor has new property AnalogColor.CommonAttribute. By default this is set to 1 and all instances of the AnalogColor is connected to the same attribute. If CommonAttribute is set to 0, one attribute can be specified for each instance and the instances can be connected to different attributes.

### ***Crossreferences options in configurator settings***

Two options in the configurator settings are added for Crossreference Simulation and Crossreference Graph. If Crossreference Simulation is set, simulation object will be added to the crossreference list. If Crossreference Graph is set, references in Ge graphs will be included in the list.

## ***Xtt commands to set signal invert, conversion and test***

Previously it was possible to set signal invert, conversion and test with commands in `rt_rtt`. Now this is also possible to do in Xtt. The new Xtt commands are

```
xtt> set signal conversion /on [/name=]
xtt> set signal conversion /off [/name=]
xtt> set signal invert /on [/name=]
xtt> set signal invert /off [/name=]
xtt> set signal test /on [/name=]
xtt> set signal test /off [/name=]
xtt> set signal testvalue /on [/name=]
xtt> set signal testvalue /off [/name=]
```

Conversion can be set on Di, Ii and Ai.

Invert can be set on Di and Do.

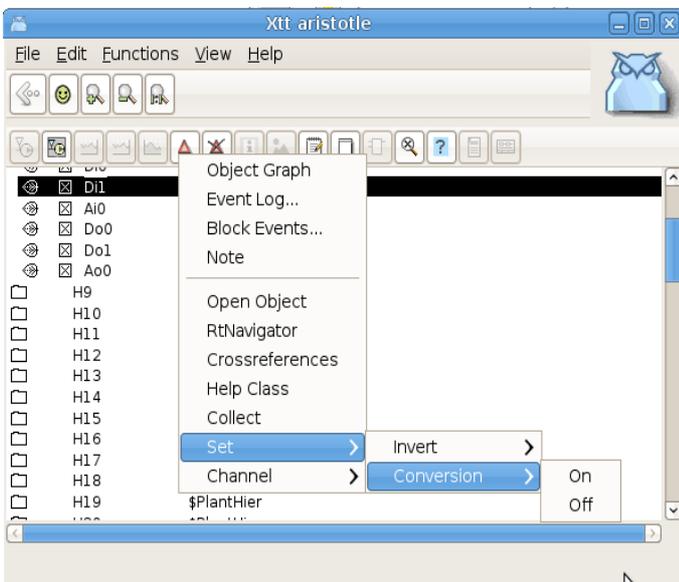
Test can be set on Do, Ao and Io.

Testvalue can be set on Do.

System privilege is required to execute the commands.

## ***Xtt menu methods to set signal, invert and test***

For Di and Do invert conversion and test can also be set from the popup menu. The new menu items for Di is Set/Conversion On/Off and Set/Invert On/Off, and the new items for Do is Invert On/Off, Set/Test On/Off and Set/TestValue True/False. The menu items are only viewed for users with System privilege.



**Fig Menu items for Set Invert and Set Conversion**

## ***Show all toplevel objects in the configurator***

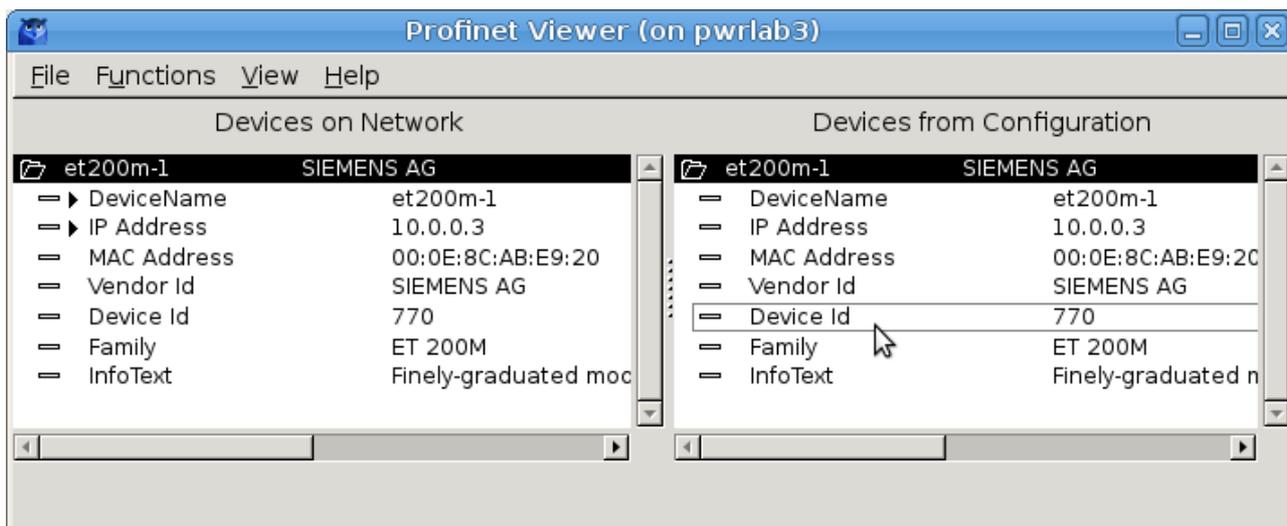
With the new menu item in the Configurator 'View/View all toplevel' the whole object hierarchy is displayed in one window, or actually in all visible windows. This function already existed before with the command 'set alltoplevel' and is now also available in the menu.

## ***Display of Profinet configuration in Xtt***

The Profinet configuration can now be displayed in Xtt by activating Show Configuration in the popup menu for a Profinet device. The xml-files for the Profinet configuration has to be distributed to \$pwrp\_load.

## ***Profinet viewer displays the configured devices***

The profinet viewer shows the Profinet devices found on the network. It can now also display the configured devices to make it easier to identify the devices on the network and download the correct device name and address. The data-file for the configuration has to be distributed to the process station. It's named \$pwrp\_load/pwr\_pnviewer\_'ethernet-device'.dat, e.g. \$pwrp\_load/pwr\_pnviewer\_eth1.dat.



**Fig Profinet viewer**

## ***Plotgroup objects in OpPlace.FastAvail***

It's now possible to insert PlotGroup objects in OpPlace.FastAvail and they can then be opened from a button in the Operator window. The new attribute ButtonText is added in the PlotGroup object to specify the text of the button.

## ***Conversion functions cdh\_XxxToString() modified***

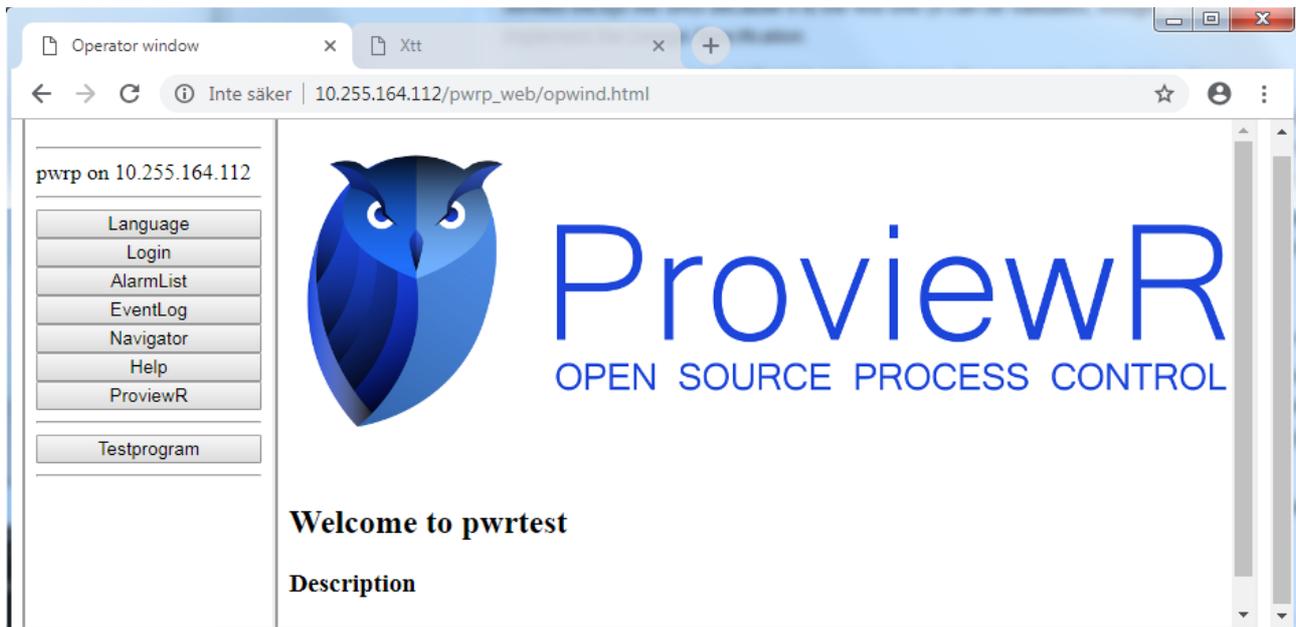
The functions below were not thread safe. They are now thread safe, but in some cases requires that an optional string argument is supplied to be thread safe. They all have modifications in the argument list or return type.

```
cdh_MaskToBinaryString()  
cdh_ClassIdToString()  
cdh_ObjectIxToString()  
cdh_ArefToString()  
cdh_TypeIdToString()  
cdh_VolumeIdToString()  
cdh_SubidToString()  
cdh_DlidToString()
```

cdh\_OidToString() is a new thread safe version of cdh\_ObjidToString().

## Web interface in HTML5 and javascript

A beta version of a new web interface written in HTML5 and javascript is introduced. The interface is runnable on most web browsers.



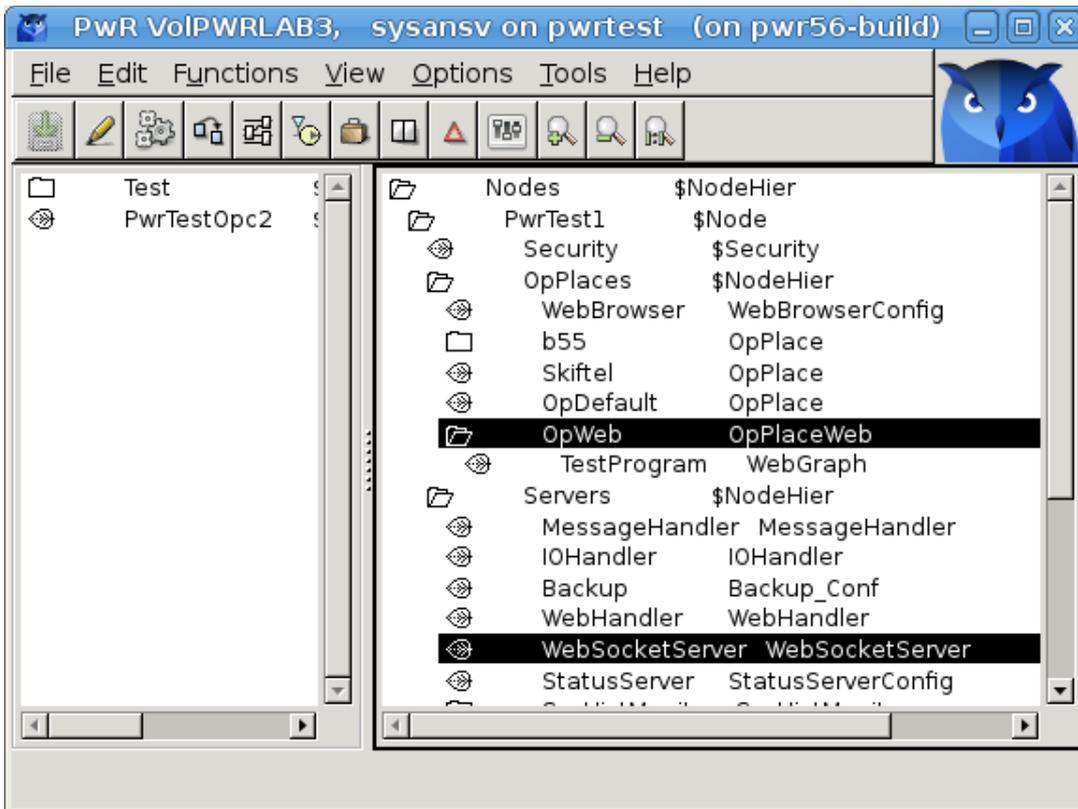
**Fig New web interface**

The following functions are implemented so far

- Operator window with menu and display of help text.
- Runtime navigator.
- Plc trace.
- Display of cross references.
- Ge graphs.
- Login.

## Configuration

- The operator menu is configured as before with a OpPlaceWeb object and below this WebGraph and WebLink objects.
- Also a server object is required, WebSocketServer.
- All files on \$pwrp\_web should be copied to \$pwrp\_web.
- To display plc trace, copy flw-files from \$pwrp\_load to \$pwrp\_web.
- To display Ge graphs, copy pwg-files from \$pwrp\_exe to \$pwrp\_web.
- To display object graphs for base classes, copy pwg-files from \$pwr\_exe to \$pwrp\_web.



**Fig OpPlace and WebSocketServer configuration**

The URL for the start page is `http://'nodename'/pwrp_web/opwind.html`

## New Types and Classes

### CurveLayoutMask

Type to configure the layout of the curve window.

### TimeRangeEnum

Type to configure default time range in the curve window.

### SevPlotGroup

Curve display on a storage station.

### SevItem, SevItemInteger, SevItemFloat and SevItemBoolean

Objects created on a storage station to monitor storage items.

### DataArithmT and DataArithmTL

The function is the same as the DataArithm object. The difference is that the DataArithmT displays a descriptive text in the function object instead of the code.

## **CompPosit**

New positioner.

## **WebSocketServer**

Server for the HTML/javascript web interface.

# **Modified Classes**

## **PlotGroup**

Attributes ButtonText, TimeRange and Layout added.

## **Di**

Added Xtt methods Invert On/Off and Conversion On/Off.

## **Do**

Added Xtt methods Invert On/Off, Test On/Off and TestValue True/False.

# **Upgrade procedure**

The upgrading has to be done from any V5.5. 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->V4.5.0->V4.6.0->V4.7.0->V4.8.6->(V5.0.0)->V5.1.0->V5.2.0->V5.3->V5.4->V5.5->V5.6

The upgrade procedure is to dump the database with reload.sh, change the version of the project in the projectlist, and then execute the script upgrade.sh.

### **NOTE !!**

Do not activate Update Classes.

If the previous version should be kept, first make a copy of the project.

## ***Make a copy of the project***

Do `sd f` to the project and start the administrator

```
> pwra
```

Now the Projectlist is opened. Enter edit mode, login as administrator if you lack access. Find 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. Save and close the administrator.

## ***Dump the databases***

Execute the first pass, *dumpdb*, in the script *reload.sh*.

```
> reload.sh
```

```
reload.sh  Dump and reload of database.
```

```
Arguments  Database or databases to reload.  
           I no arguments is supplied, all databases will be  
           reloaded.
```

```
Pass
```

```
dumpdb      Dump database to textfile $pwrp_db/'volume'.wb_dmp  
classvolumes Create structfiles and loadfiles for classvolumes  
renamedb    Rename the old database  
loaddb      Load the dump into the new database  
compile     Compile all plcprograms in the database  
createload  Create new loadfiles.  
createboot  Create bootfiles for all nodes in the project.
```

```
-- Reloading volume  directory volopg2
```

```
Pass: dumpdb classvolumes renamedbloaddb compile createload createboot
```

```
Enter start pass [dumpdb] >
```

```
-----  
Pass dump database  
-----
```

```
Do you want to continue ? [y/n/go] y  
ls: cannot access /data0/pwrp/opg2/common/db/*.wb_dmp: No such file or  
directory  
Dumping volume directory in /data0/pwrp/opg2/common/db/directory.wb_dmp  
...  
I Database opened /data0/pwrp/opg2/common/db/volopg2.db  
ls: cannot access /data0/pwrp/opg2/common/db/*.wb_load: No such file or  
directory
```

```
-----  
Pass create structfiles and loadfiles for classvolumes  
-----
```

```
Do you want to continue ? [y/n/go] n  
setdb is obsolete  
>
```

Check that the one dumpfile is created for every rootvolume

```
> cd $pwrp_db  
> ls -l *.wb_dmp
```

-rw-rw-r-- 1 cs pwrp 7467 2010-03-26 16:32 volopg2.wb\_dmp

## ***Linux release upgrade***

If you are using an older Ubuntu version upgrade the linux release and install the pwr55 package.

## ***Change version***

Enter the administrator and change the version of the project to V5.5.0. Save and close the administrator.

## ***upgrade.sh***

Do sdf to the project.

upgrade.sh is a script that is divided into a number of passes. After each pass you have to answer whether to continue with the next pass or not.

Start the script with

```
> upgrade.sh
```

Start from the savedirectory pass.

```
Enter start pass [savedirectory] >
```

### ***savedirectory***

Save the directory volume.

### ***classvolumes***

Create loadfiles and structfiles for the class volumes.

### ***cnvdump***

Convert the dumpfiles.

### ***renamedb***

Store the old databases under the name \$pwrp\_db/'volumename'.db.1.

### ***loaddb***

Create databases and load the dumpfiles into them.

### ***cnvobjects***

Convert SevNodeConfig configurations.

### ***compile***

Compile all the plc programs.

## **createload**

Create loadfiles for the root volumes.

## **createboot**

Create bootfiles for all nodes in the project.

If the project contains any application programs, these has to be built manually.

Delete files from the upgrading procedure:

```
$pwrp_db/*.wb_dmp.*
```

```
$pwrp_db/*.db.1 (old databases, directories which content also should be removed)
```

## **List example**

```
>
```

```
> sdf opg2
```

```
Setting base /data0/x5-6-0/rls
```

```
>
```

```
> upgrade.sh
```

```
upgrade.sh Upgrade from V5.5 to V5.6
```

Pass

savedirectory	Save directory volume.
classvolumes	Create loadfiles for classvolumes.
cnvdump	Convert dumpfiles.
renamedb	Rename old databases.
loaddb	Load dumpfiles.
cnvobjects	Convert objects.
compile	Compile all plcprograms in the database
createload	Create new loadfiles.
createboot	Create bootfiles for all nodes in the project.
createpackage	Create distribution packages for all nodes in the project.

```
-- Upgrade opg2
```

```
Enter start pass [savedirectory] >
```

```
-----  
Pass save directory volume  
-----
```

```
Do you want to continue ? [y/n/go] y
```

```
-- Processing line: 270
```

```
-- Building volume directory
```

```
I Volume directory loaded
```

```
I Database opened /data0/pwrp/pwrtestloc/src/db/directory.wb_load
```

```
-- Saving file /data0/pwrp/pwrtestloc/src/db/directory.wb_load ->  
/data0/pwrp/pwrtestloc/src/db/directory.wb_load.1
```

```
%WNAV-E-MSG, Session saved
```

-----  
Pass create structfiles and loadfiles for classvolumes  
-----

Do you want to continue ? [y/n/go] y  
ls: cannot access /data0/pwrp/opg2/src/db/\*.wb\_load: No such file or directory  
-----

Pass convert dumpfiles  
-----

Do you want to continue ? [y/n/go] y  
/data0/pwrp/opg2/src/db/volopg2.wb\_dmp  
-----

Pass rename old databases  
-----

Do you want to continue ? [y/n/go] y  
-- Saving file /data0/pwrp/opg2/src/db/volopg.db ->  
/data0/pwrp/opg2/src/db/volopg.db.1  
-----

Pass load database  
-----

Do you want to continue ? [y/n/go] y  
-- Loading volume volopg  
...  
-- Processing line: 57  
-- Building volume directory  
I Volume directory loaded  
I Database opened /data0/pwrp/opg2/src/db/directory.wb\_load  
-- Processing line: 200  
-- Building volume Vol0pg  
I Volume Vol0pg loaded  
Berkeley DB 4.6.21: (September 27, 2007)  
info put: 0  
Berkeley DB 4.6.21: (September 27, 2007)  
info get: 0  
int rc = m\_txn->abort(): 0  
-----

Pass convert objects in loaded database  
-----

Do you want to continue ? [y/n/go] y  
-- Processing line: 200  
-- Building volume directory  
I Volume directory loaded  
I Database opened /data0/pwrp/opg2/src/db/directory.wb\_load  
-- Saving file /data0/pwrp/opg2/src/db/directory.wb\_load ->  
/data0/pwrp/opg2/src/db/directory.wb\_load.1  
Print wbl  
-- Writing line: 200  
E Error, Volume 'Vol0pgSev' is not yet created.  
%WNAV-E-MSG, Session saved  
-----

Pass compile plcprograms  
-----

Do you want to continue ? [y/n/go] y

...

Berkeley DB 4.6.21: (September 27, 2007)

info get: 0

I Database opened /data0/pwrp/opg2/src/db/volopg.db

```
-- Plc window generated          F1-Z1-Plc-W
-- Plc window compiled for x86_linux optimized -03 F1-Z1-Plc-W
-- Plc plcpgm compiled for x86_linux optimized -03 F1-Z1-Plc
-- Plc window generated          F1-Z2-Plc-W
-- Plc window compiled for x86_linux optimized -03 F1-Z2-Plc-W
-- Plc plcpgm compiled for x86_linux optimized -03 F1-Z2-Plc
```

-----  
Pass create loadfiles  
-----

Do you want to continue ? [y/n/go] y

-- Removing old loadfiles

rm: cannot remove `/data0/pwrp/opg2/bld/common/load/ld\_vol\*.dat': No such file or directory

...

Berkeley DB 4.6.21: (September 27, 2007)

info get: 0

I Database opened /data0/pwrp/opg2/src/db/volopg.db

```
-- Building archive for volume: 000_001_001_012
-- Archive built for volume: 000_001_001_012
```

```
-- Working with load file volume 'Vol0pg'...
```

```
-- Open file...
```

```
-- Successfully created load file for volume 'Vol0pg'
```

```
-- 26 objects with a total body size of 21976 bytes were written to new file.
```

Before this pass you should compile the modules included by ra\_plc\_user.

-----  
Pass create bootfiles  
-----

Do you want to continue ? [y/n/go] y

-- Creating bootfiles for all nodes

ProviewR is free software; covered by the GNU General Public License. You can redistribute it and/or modify it under the terms of this license.

ProviewR is distributed in the hope that it will be useful but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

```
-- Creating bootfile for node opg
```

```
    plc_opg_0507_00011
```

```
-- Plc thread generated priority 0, scantime 0.10000 s, 2 plcpgm's
```

```
-- Plc process compiled for x86_linux optimized -03 Dummy
```

```
-- Plc program linked for x86_linux node plc_opg_0507
```

```
-- Creating bootfile for node aristotle
```

```
    plc_aristotle_0517_00011
-- Plc thread generated priority 0, scantime 0.10000 s, 2 plcpgm's
-- Plc process compiled for x86_linux optimized -O3 Dummy
-- Plc program linked for x86_linux node plc_aristotle_0517

-- The upgrade procedure is now accomplished.
```

```
setdb is obsolete
```

```
>
```

```
>
```