Release Notes V4.8

2012 05 16

Copyright SSAB Oxelösund AB 2010

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 Proview V4.8.0....6
New functions....6
Profinet....6
Web interface....6
Velleman K8055 experiment board....7
History curve window....7
Xtt logging condition....7
Operator log....7
Xtt commands....8
oplog....8
UTF-8 coded translation tables and help file texts....8
Embedded Linux....8
ARM architecture....8
Mac OS X....8
New Classes....8
PnControllerSoftingPNAK....8
PnDevice....8
PnModule....9
Siemens_ET200S_PnDevice....9
Siemens_ET200M_PnDevice....9
Sinamics_G120_PnDevice....9
ABB_ACS_PnDevice....9
BaseFcPPO5IoModule....9
BaseFcPPO3IoModule....9
BaseFcPPO3PnModule....9
Sinamics_Tgm1_PnModule....9
Siemens_Ai2_PnModule....9
Siemens_Ao2_PnModule....9
Siemens_Di4_PnModule....10
Siemens_Di2_PnModule....10
Siemens_Do4_PnModule....10
Siemens_Do2_PnModule....10
Siemens_Do32_PnModule....10
Siemens_D16_PnModule....10
Siemens_Do8_PnModule....10
Siemens_Di32_PnModule....10
Siemens_Di16_PnModule....10
Siemens_Di8_PnModule....10
Siemens_Dx16_PnModule....10
Siemens_Ai8_PnModule....10
Siemens_Ao8_PnModule....10
Siemens_Ai4_PnModule....11
Siemens_Ao4_PnModule....11
Sinamics_G120_Tgm1....11
Sinamics_G120_Tgm1Fo....11
GPIO....11
GPIO_Module 11
OneWire....11
Maxim_DS18B20....11
USB_Agent....11

Velleman_K8055....11
Velleman_K8055_Board....11
Modified Classes....12
BaseFcPPO5PbModule....12
BaseFcPPO3PbModule....12
BaseFcPPO3....12
V4.8.1 Additions....12
Remote support for WebSpheare Message Queue....12
I/O support for Arduino USB boards....12
PID controller derivative filter algorithm modified....12
Logging of events in process graphics....12
Upgrade from V4.8.0 to V4.8.1....12
V4.8.2 Additions....13
Miscellanous module and class volume....13
Hilscher cifX board....13
UDP IO....13
USB Joystick....13
NMps cells with size 60 and 120....14
Axis arc....14
Upgrade from V4.8.0 or V4.8.1 to V4.8.2....14
V4.8.3 Additions....14
Post server....14
Report server....16
Xtt Collect window....18
Show all signals in a Collect window....18
Ge multiline texts....18
Web site opened from other location....19
New Plc objects....19
Modbus RTU....19
V4.8.4 Additions....19
Buffer channels and signals....19
Buffer Inputs....19
Buffer Outputs....20
ChanBiBlob and ChanBoBlob....20
Sev database engine....20
V4.8.5 Additions....21
ConversionOn, InvertOn, TestOn displayed in rt_xtt....21
Ge pie chart....21
Ge bar chart....21
Custom logotype....22
Trend curves....22
Configuration....24
Plc compile with warnings....24
Name changed of option file and plc exectuable....24
Multiple plc processes with CPU affinity....25
V4.8.6 Additions....25
Pie chart and bar chart in java....25
Java Trend and Fast curves....25
Class editor function object copy....25
Plc object DataArithmL added....25
Xtt operator log configured in OpPlace object....25
Disable Qcom links between all nodes in a project....25
Display values in Xtt as hex, octal and binary....26
Several plc processes in System Status graph....26

Print dialog....26
Plc trace version control....27
Volume clones....27
Simulate server....28
Sev database statistic table added....29
Sev database with SQLite....29
Ge Axis dynamics....29
Upgrade from V4.8.X to V4.8.6....30
Upgrade procedure 30
Make a copy of the project....30
Dump the databases....30
Linux release upgrade....31
Change version....31
upgrade.sh....31
classvolumes....32
renamedb....32
cnvdump....32
loaddb....32
compile....32
createload....32
createboot....32
List example....32
Appendix A Embedded Linux....36
Project....37
Installing Proview....38
Proview runtime....38
Project....40
Settings....40
Distribute....40
Appendix B Mac OS X....41
Build from source on Mac OS X 10.6 x86_64....41

Upgrading to Proview V4.8.0

This document describes new functions i Proview V4.8.0, and how to upgrade a project from V4.7.0 to V4.8.0.

New functions

Profinet

An interface to the Softing profinet stack PNAK is implemented in V4.8. Also an Profinet configurator using the gsdml configuration files is added. Also, using the same Profinet stack, there exist a new tool for identifying devices on a Profinet network and function to set name and ip-address for the repective devices. Read more about how th use the Profinet interface in *"Guide to I/O system*".

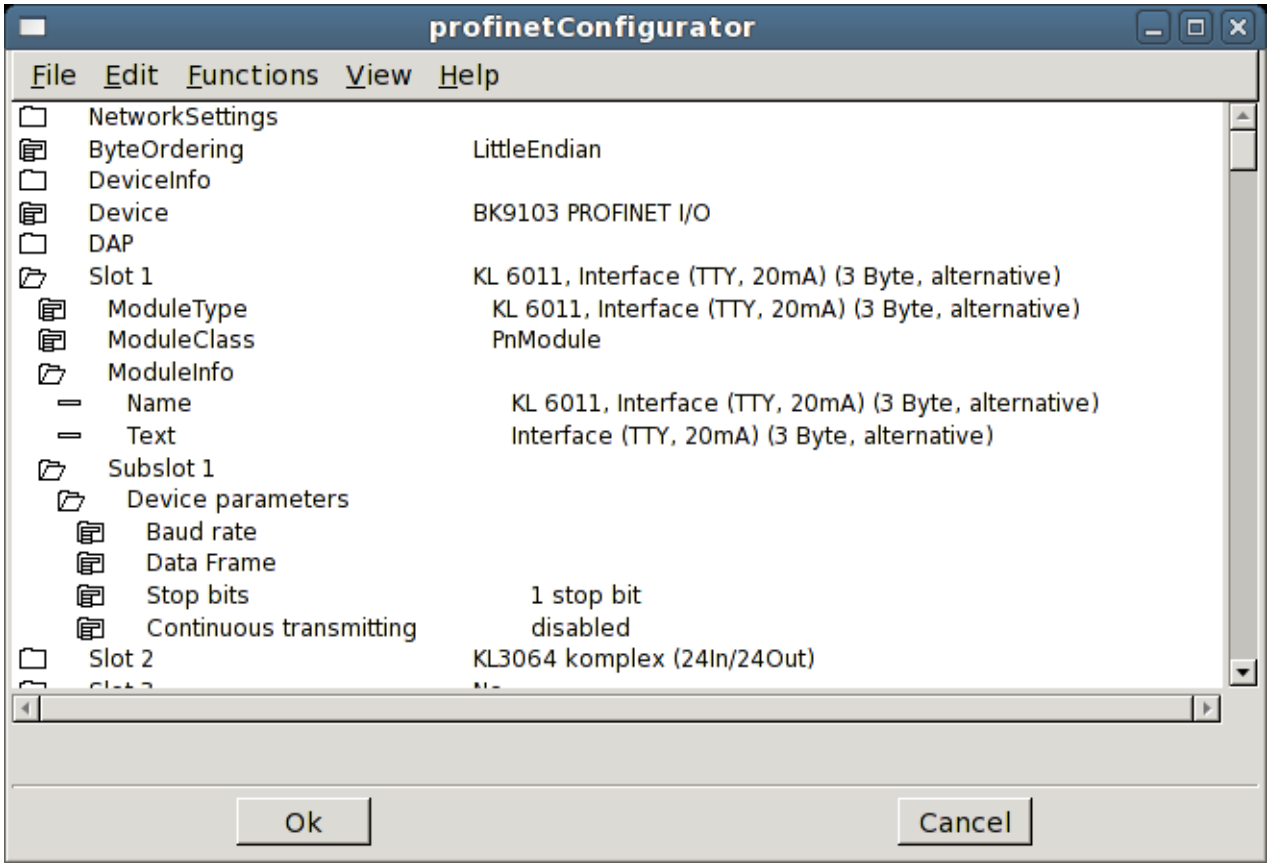


Fig The Profinet configurator

Web interface

- Language support is added to the Web interface.
- Method buttons in object graph added.
- Object graphs updated.

Velleman K8055 experiment board

Velleman K8055 is an USB experiment board with 2 Ai, 5 Di, 8 Do and 2 Ao. It is can be purchased as a kit, K8055, or as an assembled board, VM110. The card can be used to test Proview with some simple application.

The board is configured with the objects USB_Agent, Velleman_K8055 and K8055_Board. See Guide to I/O System for more info.

History curve window

The curve window for process history has a new toolbar to choose time interval.

It is also possible view curves for several stored attributes, either by configuring a PlotGroup object containing references to a number of SevHist objects, or by adding curves to a curve window.

You add a curve by selecting the signal, or SevHist object in the navigator and press the '+' button.

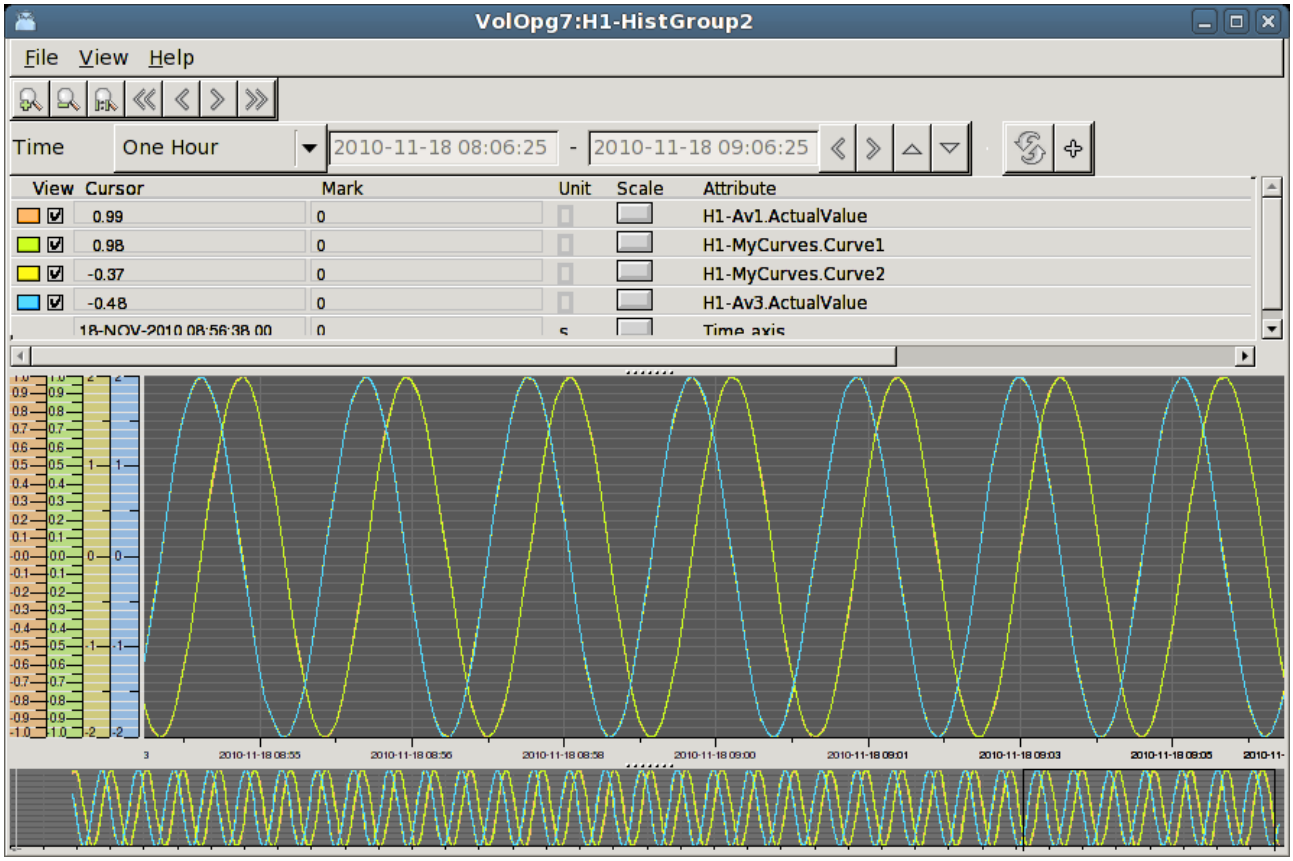


Fig History curve window with multiple curves

Xtt logging condition

The condition property of the Xtt logging function can contain an expression where values of database attributes are fetched by the GetD, GetA and GetI functions for digital, analog and integer attributes respectively, e.g

```
expr( GetD("H1-Dv1.ActualValue") && GetA("H1-Av3.ActualValue") > -0.8)
```

Operator log

Operator actions can be logged to file.

The logging is started with the xtt command 'oplog start' and the default log file is

$pwrp_log/xtt.log. The logging is stopped with the command 'oplog stop' .

A log file can be played, i.e. all actions in the file are executed, with the command 'oplog play'.

Xtt commands

oplog

New command to handle operator log.

xtt> oplog start [/file=]	Start the logging.
xtt> oplog stop	Stop the logging.
xtt> oplog play [/file=] [/speed=]	Execute logged actions.

UTF-8 coded translation tables and help file texts

It is possible to write UTF-8 coded translations tables and help file texts by adding the 'Coding:UTF-8' tag on the first line. This will make it possible to create translated versions of the operator environment to most languages. A translation to chinese (zh_cn) is planned to the next release.

Embedded Linux

Support to build Proview for embedded Linux with cross compilation is added to V4.8.0. See Appendix A.

ARM architecture

Build files for ARM are added to the runtime module. To build for ARM see Appendix A.

Mac OS X

Proview can be built on Mac OS X 10.6 on x86_64 by using fink. See Appendix B for more info.

New Classes

PnControllerSoftingPNAK

I/O Agent object configuring the Softing Profinet stack PNAC.

PnDevice

I/O Rack object configuring a Profinet device. The device is configured by the Profinet configurator opened from the ConfigureDevice method.

PnModule

I/O Card object configuring a Profinet module. The module objects are created by the Profinet configurator.

Siemens_ET200S_PnDevice

Device object for a Siemens ET200S.

Siemens_ET200M_PnDevice

Device object for a Siemens ET200M.

Sinamics_G120_PnDevice

Device object for a Sinamics G120 drive.

ABB_ACS_PnDevice

Device object for a ABB ACS800 drive with RETA-02 interface (profinet).

BaseFcPPO5IoModule

Generic I/O Bus module object for PPO5. Contains the Channels for the communication and should be an attribute object of the module object for the IO bus system.

BaseFcPPO3IoModule

Generic I/O Bus module object for PPO3 / Standard telegram 1. Contains the Channels for the communication and should be an attribute object of the module object for the IO bus system.

BaseFcPPO3PnModule

Module object for a drive using Standard Telegram 1 / PPO3. The Io-attribute of this object can be directly connected to a drive object of type BaseFcPPO3 in the $PlantHier. This one in turn can be connected to a function object in a plc-program of type BaseFcPPO3Fo.

Sinamics_Tgm1_PnModule

Module object for a drive using Standard Telegram 1. The Io-attribute of this object can be directly connected to a drive object of type Sinamics_G120_Tgm1 in the $PlantHier. This one in turn can be connected to a function object in a plc-program of type Sinamics_G120_Tgm1Fo.

Siemens_Ai2_PnModule

Module object for a Siemens ET200 module with 2 analog inputs.

Siemens_Ao2_PnModule

Module object for a Siemens ET200 module with 2 analog outputs.

Siemens_Di4_PnModule

Module object for a Siemens ET200 module with 4 digital inputs.

Siemens_Di2_PnModule

Module object for a Siemens ET200 module with 2 digital inputs.

Siemens_Do4_PnModule

Module object for a Siemens ET200M module with 4 digital outputs.

Siemens_Do2_PnModule

Module object for a Siemens ET200 module with 2 digital outputs.

Siemens_Do32_PnModule

Module object for a Siemens ET200 module with 32 digital outputs.

Siemens_D16_PnModule

Module object for a Siemens ET200 module with 16 digital outputs.

Siemens_Do8_PnModule

Module object for a Siemens ET200 module with 8 digital outputs.

Siemens_Di32_PnModule

Module object for a Siemens ET200 module with 32 digital inputs.

Siemens_Di16_PnModule

Module object for a Siemens ET200 module with 16 digital inputs.

Siemens_Di8_PnModule

Module object for a Siemens ET200 module with 8 digital outputs.

Siemens_Dx16_PnModule

Module object for a Siemens ET200 module with 16 digital outputs and 16 digital inputs.

Siemens_Ai8_PnModule

Module object for a Siemens ET200 module with 8 analog inputs.

Siemens_Ao8_PnModule

Module object for a Siemens ET200 module with 8 analog outputs.

Siemens_Ai4_PnModule

Module object for a Siemens ET200 module with 4 analog inputs.

Siemens_Ao4_PnModule

Module object for a Siemens ET200 module with 4 analog outputs.

Sinamics_G120_Tgm1

Class representing a G120 drive using Standard telegram 1. The I/O can be directly connected to a BaseFcPPO3IoModule.

Sinamics_G120_Tgm1Fo

Function object working on a Sinamics _G120_Tgm1 drive object. Connect drive to function object with the connect method.

GPIO

I/O Rack object configuring GPIO, General Purpos I/O.

GPIO_Module

I/O Card object configuring GPIO.

OneWire

I/O Rack object configuring the Maxim 1-wire bus.

Maxim_DS18B20

I/O object configuring the Maxim DS18B20 temperature sensor on the 1-wire bus.

USB_Agent

I/O Agent object initializing libusb for attachment of USB devices.

Velleman_K8055

I/O Rack object for Velleman K8055 experiment board.

Velleman_K8055_Board

I/O Card object for Velleman K8055 experiment board.

Modified Classes

BaseFcPPO5PbModule

The internal channels are moved to an internal BaseFcPPO5IoModule attribute object.

BaseFcPPO3PbModule

The internal channels are moved to an internal BaseFcPPO3IoModule attribute object.

BaseFcPPO3

Added attributes for Status Word and Control Word as bitmasks.

V4.8.1 Additions

Remote support for WebSpheare Message Queue

WebSpheare Message Queue is implementet as a remote communication protocol. Configure with a RemnodeWMQ object.

I/O support for Arduino USB boards

Arduino USB boards are configured with the rack object Ardunio_USB and the card object Arduino_UNO. See Guide to I/O Systems for more info.

PID controller derivative filter algorithm modified

The filter algorithm for the derivative part of the PID controller is modified.

Logging of events in process graphics

All events in the process graph, such as cursor motion, button clicks, keyboard actions, can be logged on file, and replayed to repeat the actions. This is a usable function for debugging the operator environment. It is started by the xtt command 'oplog start /event/file='.

Upgrade from V4.8.0 to V4.8.1

Enter the administrator and change the version of the project to V4.8.1. Save and close the administrator.

I you have any class volumes, enter the class editor and build the volume.

Enter the configurator for each root volume and activate 'Function/Update Classes' and build.

V4.8.2 Additions

Miscellanous module and class volume

A new module and classvolume is added for miscellanous function. It contains a pingping game and an object that sets patterns to Do boards for demo usage.

Fig Pingpong game Misc_PingPong

Hilscher cifX board

Hilscher cifX is a family of I/O cards handling a number of different I/O buses, CANopen, CC-Link, DeviceNet, EtherCAT, EtherNet/IP, Modbus TCP, Powerlink, Profibus, Profinet and Sercos III. The interface to Proview is the same for all the boards and buses.

The board and the devices/slaves on the bus are configured in the SYCON.net configurator on Windows. The configuration is exported to a configuration file, that is copied to the process station.

Read more in Guide to IO Systems.

A beta version of an cifX implementation of Profinet controller, using the Proivew configurator, is also present.

UDP IO

The UPD IO make is possible to send and recieive UDP messages with the IO interface, ie configure the message are with input and output channels, and use signal objects to set or get the content of the message. The UDP IO is configured with the UDP_IO card object and a generic rack object BaseIORack.

USB Joystick

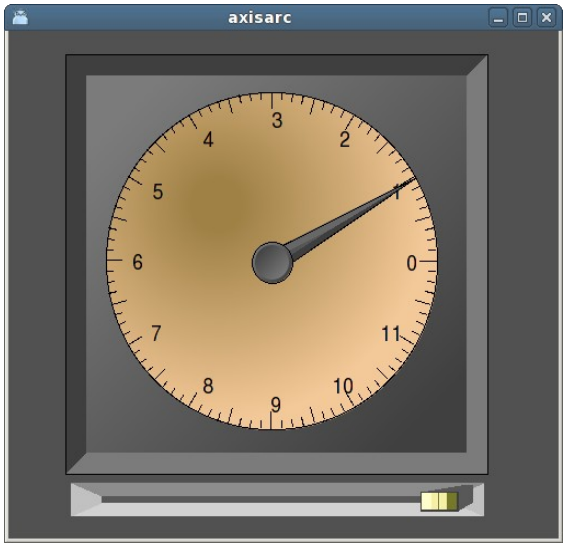
USB Joysticks are configured with the USB_Joystick object, or with a subclass of this class as the CodeMerc_JoyWarrior that is configured for a Code Mercenarias JoyWarrior with 3 axes and 8 buttons.

NMps cells with size 60 and 120

Previously only NMps cells of size 30 existed. Cells with size 60 and 120 are added where the data references are implemented as arrays.

Axis arc

Ge component AxisArc for a circular scale.



Upgrade from V4.8.0 or V4.8.1 to V4.8.2

Enter the administrator and change the version of the project to V4.8.2. Save and close the administrator.

I you have any class volumes, enter the class editor and build the volume.

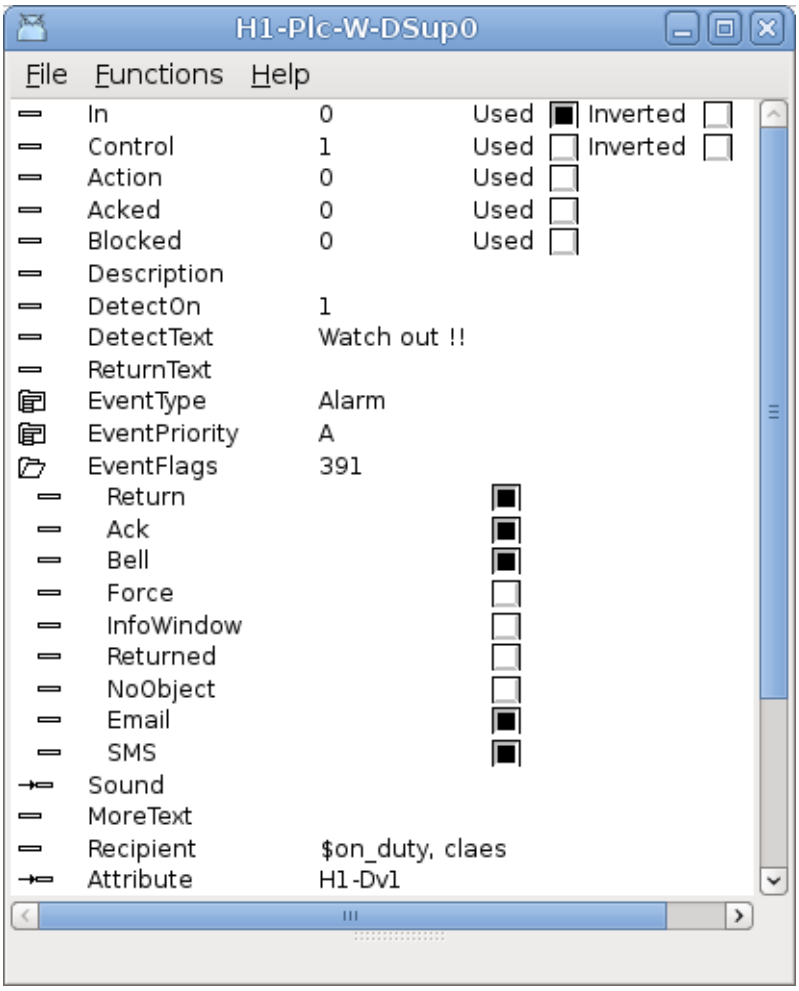
Enter the configurator for each root volume and activate 'Function/Update Classes' and build.

V4.8.3 Additions

Post server

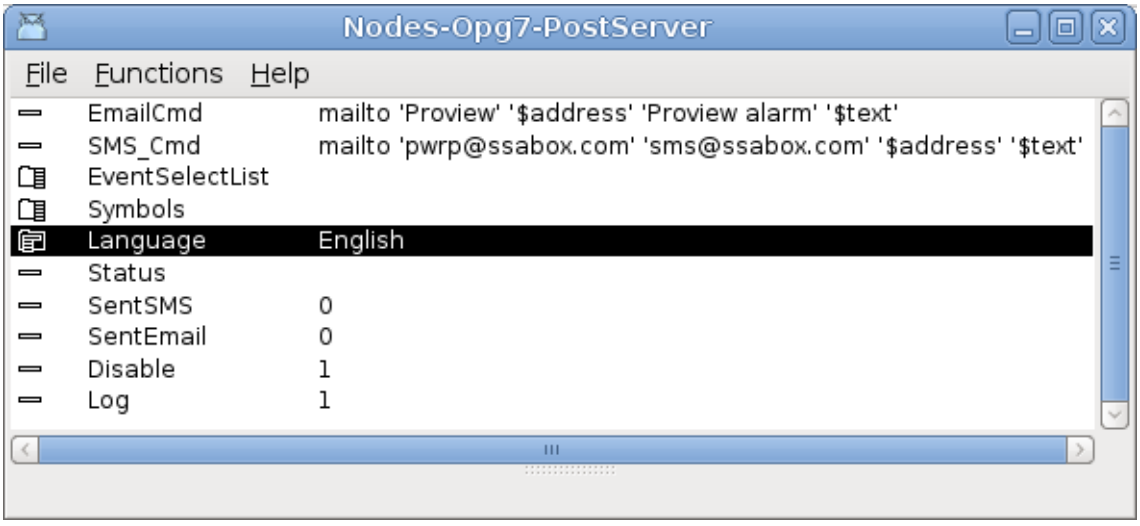
The post server sends events and alarms as epost or sms messages.

A messages is configured with a Dsup or Asup object with the bits Email or SMS in the EventFlags attribute filled in. Also a recepient has to be stated.



The recipient is a user, or a number of users, defined in the user database. The alarm text is sent to the epost adress or SMS number defnied for these users. It is also possible to define symbols that contains users, and specify the symbols with a $ sign before the symbol name. The symbols are defined in the Symbol array in the PostConfig object.

The post server is configured with a PostConfig object in the node hierarchy.



A shell command for sending the mail it inserted into the EmailCmd attribute, and also a shell command to send SMS messages is stated in SMS_Cmd attribute. The commands contatins the symbols $address and $text that will be replaced by the epost address/SMS number and the alarm

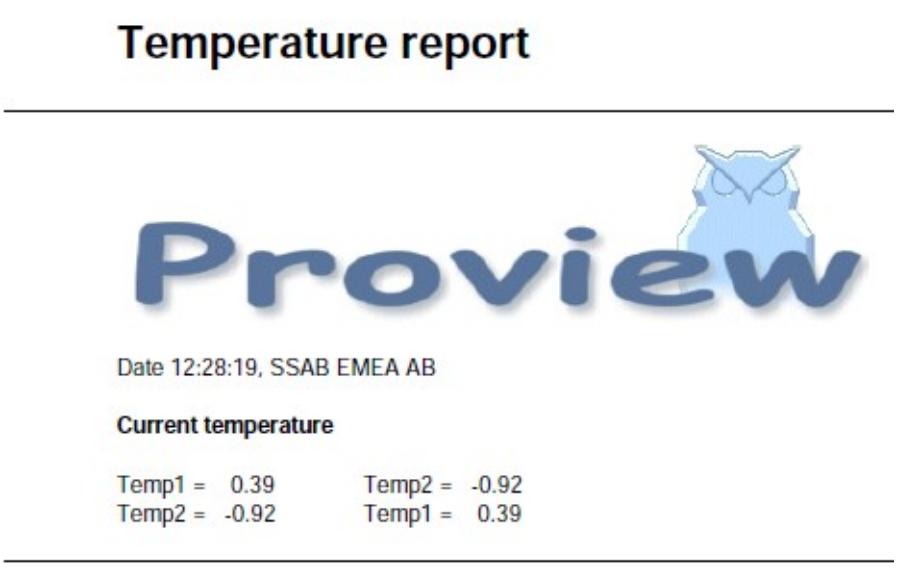
text.

Report server

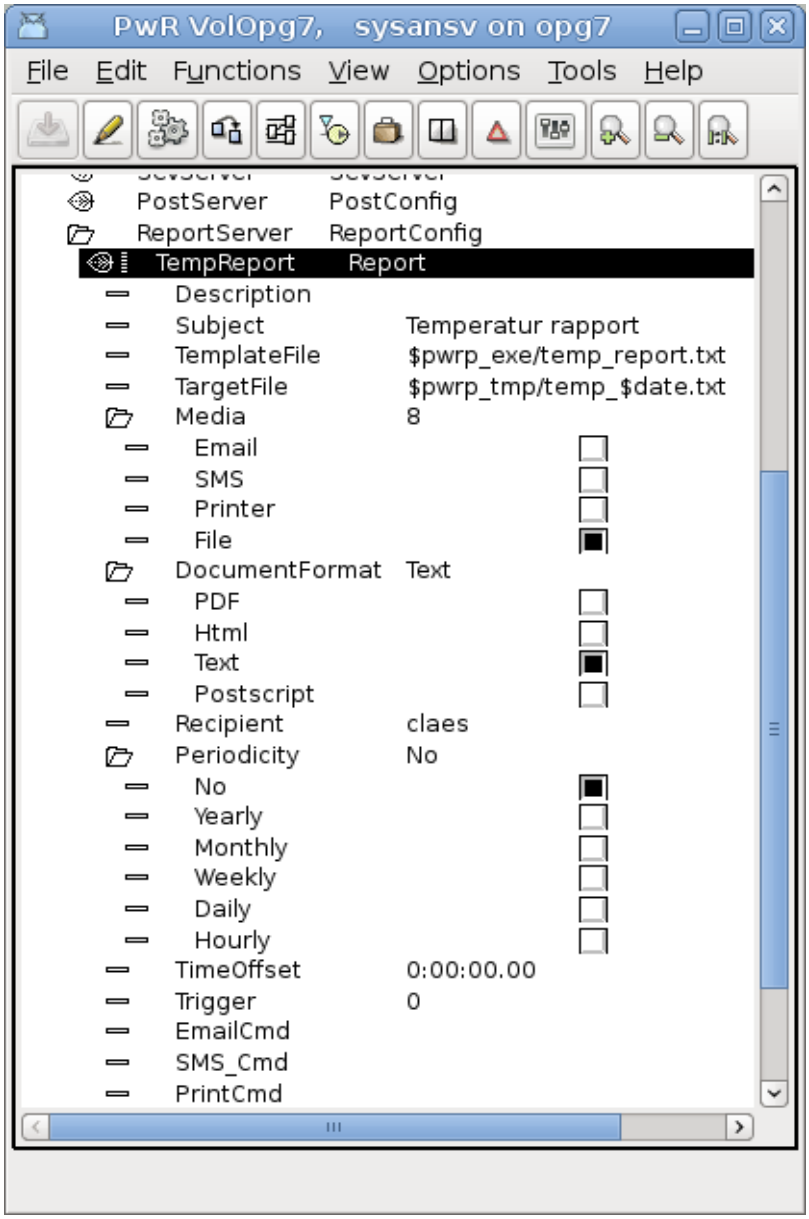
The Report server handles reports. A report contains a text that includes current values from attributes in the realtime database. The text is defined with a template of xtt help format. The value tag prints the value of the specified attribute with the specified format, eg <value attr=H1-Av1.ActualValue format=%7.2f> will print the current value of H1-Av1 with the format %7.2 which means with 7 characters and 2 decimals. Note also the date tag in the example below.

```
Temperature report
<hr>

image> pwr_logga.gif

Date <date format=%1t>, SSAB EMEA AB

<b>Current temperature

Temp1 = <value attr=H1-Av1.ActualValue format=%7.2f> <t>Temp2 = \
<value attr=H1-Av2.ActualValue format=%7.2f>
Temp2 = <value attr=H1-Av2.ActualValue format=%7.2f> <t>Temp1 = \
<value attr=H1-Av1.ActualValue format=%7.2f>

<hr>
```

This is the apperance of the above template file in pdf format.



A report is configured by a Report object.

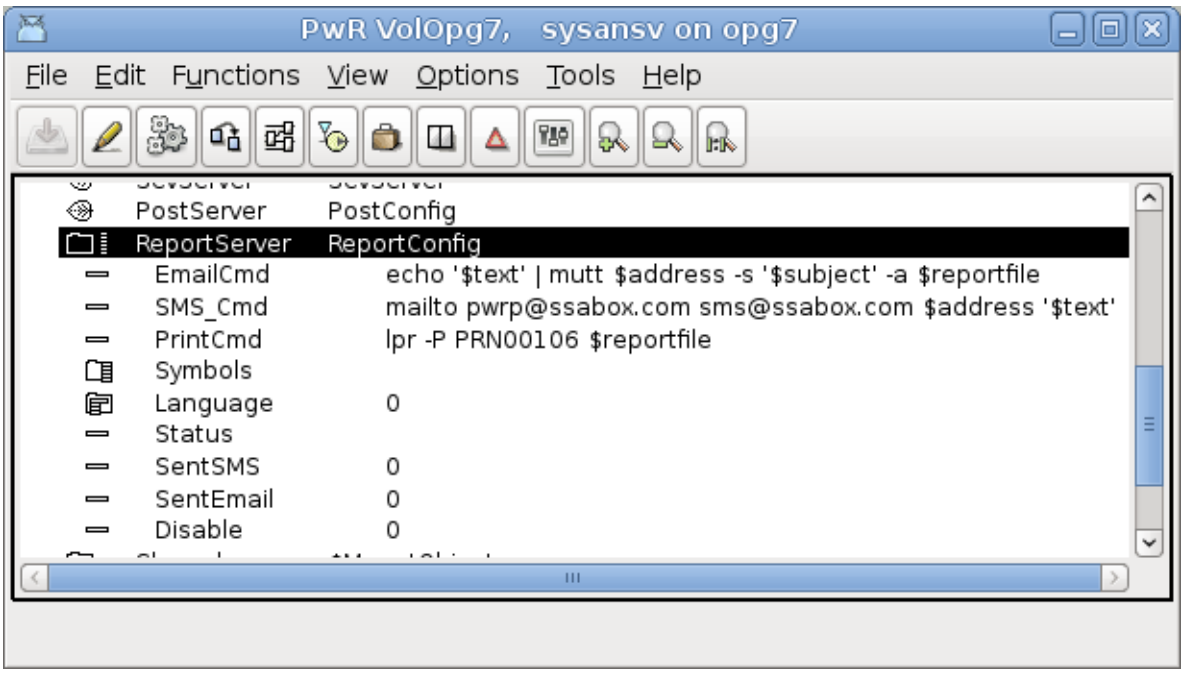


The report object is usually configured below the ReportConfig object, that configures the report server. The report can be triggered manually, or from the plc program, by setting the Trigger attribute. It can also be sent periodically by specifing how often and at what time the report should be sent or stored.

The report can be

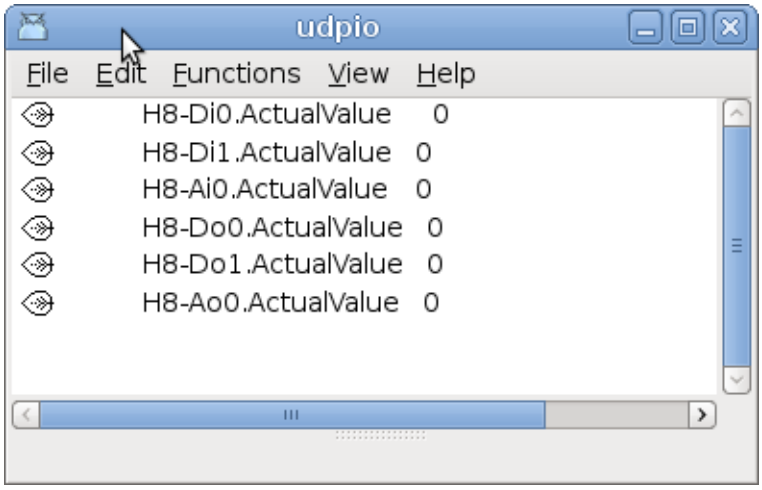
- Sent as email, as an attached pdf-file, as an html or as plain text.
- Sent as sms as plain text.
- Sent to printer as postscript.
- Written to file as pdf-file, html-file or text-file.

The report server is configured by a ReportConfig object in the plant hiearchy.



Xtt Collect window

Attributes collected with the collect function in xtt can be displayed in a separate window. Multiple collect window can be used concurrently.



The collect window can be stored and opened again later.

Show all signals in a Collect window

A new function show all signals in a hierarchy in a collect window. Select the hierarchy and activate Functions / Collect / Signals (Shift+Ctrl+T) in the xtt menu.

It is also possible to display all IO signals, ie Di, Do, Ai, Ao, Ii, Io and Co, in a hierachy in a collect window by activating Functions / Collect /IOSignals (Ctrl + T) in the xtt menu.

Ge multiline texts

Multiline texts contatined in Text attributes can be displayed in Ge.

Web site opened from other location

It is now possible to place the website no another server than the operator or process station.

In the attribute PwrHost the Proview node to attach to is stated. To be able to open a socket to the Proview node, one has to

To be able to open the web interface from another site than the process or operator station, the host name of the process station has to be stated in PwrHost and a signature has to be applied to the jar-files of the web site. The signature is an alias created by keytool and is stated in AppletSignature.

In this example a signature with the alias pwrkey is created with a validity of 100 years

```
> keytool -genkey -alias pwrkey
> keytool -selfcert -alias pwrkey -validity 36500
```

The alias should be signed to all jarfiles in $pwrp_web, pwr_rt_client.jar, pwr_jop.jar, pwr_jopc.jar etc. This is done with jarsigner

```
> jarsigner $pwrp_web/pwr_rt_client.jar pwrkey
```

The project specific jarfile will automatically be signed if the alias is stated in AppletSignature. Note that the passphrase for the keystore has to be entered in the terminal window. It is possible to state the passphrase in the AppletSignature after the alias and a '/', eg 'pwrkey/mypassphrase'.

New Plc objects

PulseTrain – generates pulstrains with cycletimes from 1 ms to 30s.

True – true output.

False – false output.

EnumToStr – convert enum value to corresponding string.

StrToEnum – convert enum string to value.

Modbus RTU

V4.8.3 contains a beta release of Modbus RTU master and slave.

V4.8.4 Additions

Buffer channels and signals

The buffer channels, ChanBo and ChanBi, and corresponding signal objects, Bo and Bi, are used for tranferring IO data of type arrays, strings or binary data.

Also the channel objects ChanBoBlob and ChanBiBlob are implemented where the IO data is not stored in an Area object and IO-copied, but copied directly to the specified target attribute.

Buffer IO is implemetned for bus IO, eg Profibus, Profinet, Modbus, IO_UDP, Libnodave etc

Buffer Inputs

Buffer inptuts are configured by placing a ChanBi object under a card object. The size of the

received data area is specified in the Size attribute. Also Representation shoud be stated.

The ChanBi is connected to a subclass of the Bi class. There are some predefined subclasses, BiIntArray20, BiIntArray100, BiFloatArray20, BiFloatArray100 and BiString80. Signal objects of other sizes can be defined as userclasses, and has to contain an ActualValue attribute and an InitValue attribute in that order. ActualValue should be and array of pointers to the desired data type, and InitValue an array of the desired datatype. For single strings though, ActualValue is a string pointer, and InitalValue a string.

To use the buffer input signal in the plc code, function object of type GetBiInt32, GetBiFloat32, GetBiString80, StoBiInt32, StoBiFloat32, StoBiString80, CStoBiInt32, CStoBiFloat32 and CStoBiString80 are used.

Buffer Outputs

Buffer outputs are configured in a similar way with ChanBo object below the card object. It is connected to a subclass of the Bo class, and the predefined Bo classes are BoIntArray20, BoIntArray100, BoFloatArray20, BoFloatArray100, BoString80.

To use the buffer input signal in the plc code, function object of type GetBoInt32, GetBoFloat32, GetBoString80, StoBoInt32, StoBoFloat32, StoBoString80, CStoBoInt32, CStoBoFloat32 and CstoBoString80 are used.

ChanBiBlob and ChanBoBlob

The normal buffer channels, ChanBi and ChanBo, are IO copied to ensure that the signal value is not changed during a plc scan. This means that the signal value is stored in area object in the System volume, and that each plc thread makes a copy of this area and reads all signal values from the copy. In cases where the IO copying are not important, and the overhead is unwanted, channels of type ChanBiBlob and ChanBoBlob can be used instead.

ChanBiBlob and ChanBoBlob does not have a corresponding signal type. Instead they are connected to an aribitrary attribute, and for inputs, the content of the IO area is copied to the attribute, while for outputs the content of the buffer attribute is copied to the IO area. There is no conversion of the content of the area content and no byte swapping for different byte ordering are performed.

Sev database engine

The database engine for the tables in the Sev mysql database can be set from the configration file /etc/proview.cnf with the sevMysqlEngine, eg

```
sevMysqlEngine          innodb
```

This will only affect new created tables. To modify existing tables sev_repair has to be executed with the option -e

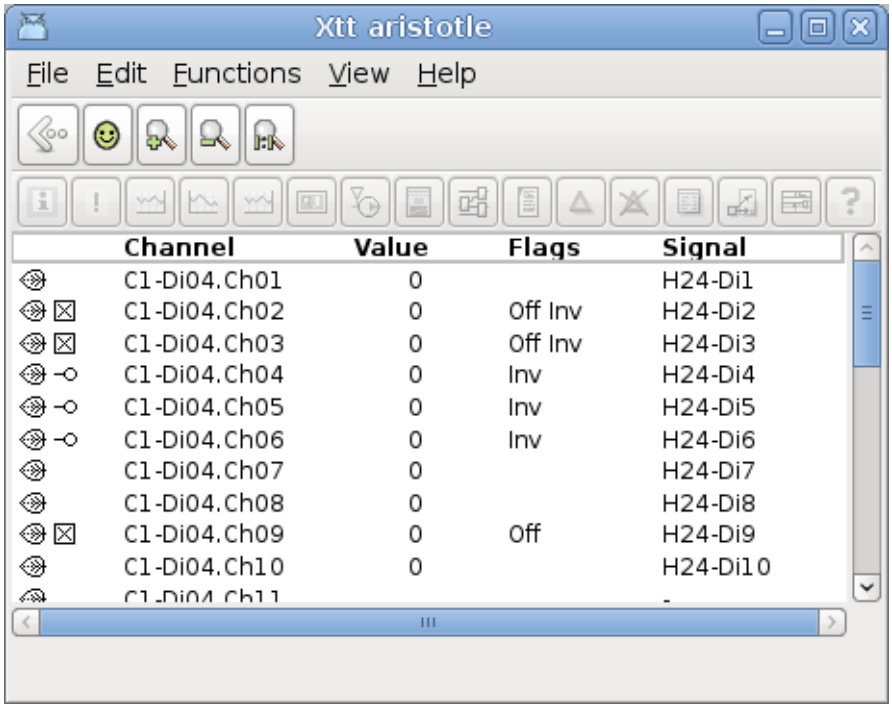
```
> sev_repair -e
```

This will set the engine of all existing tables to the engine specified in /etc/proview.cnf

V4.8.5 Additions

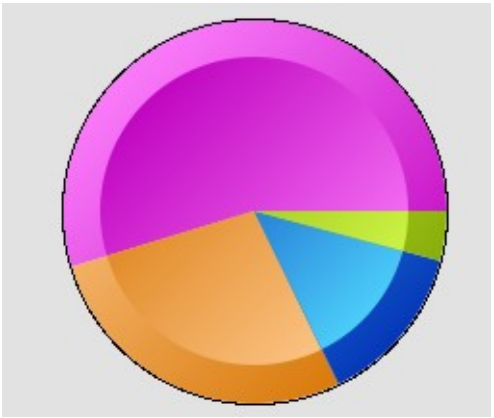
ConversionOn, InvertOn, TestOn displayed in rt_xtt

The ConversionOn, InvertOn and TestOn flags for IO channels are displayed with an icon for the channel and corresponding signal i rt_xtt.



Ge pie chart

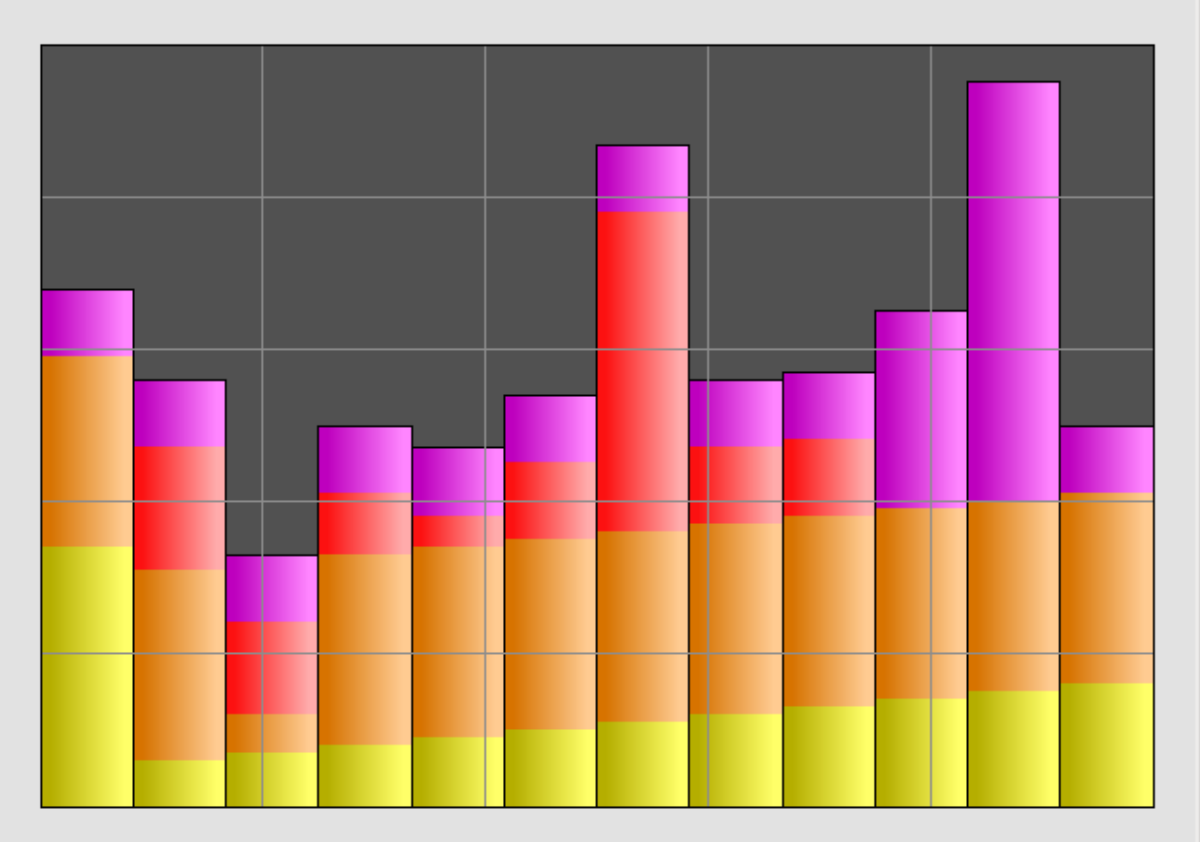
New Pie chart in Ge. The chart is found under the Analog map in the Ge palette. It can be divided into 12 sectors, and each sector is connected to an analog signal.



Ge bar chart

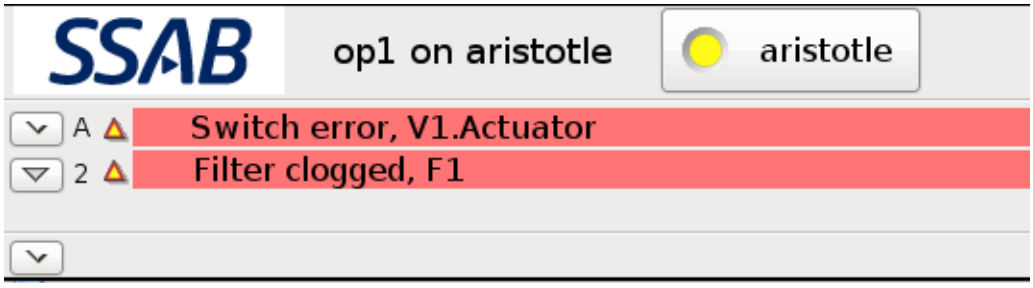
New Bar chart in Ge. Each bar can be divided into 12 segments of different color. Each segment is

connected to an array that will determine the size of the segment for the different bars.



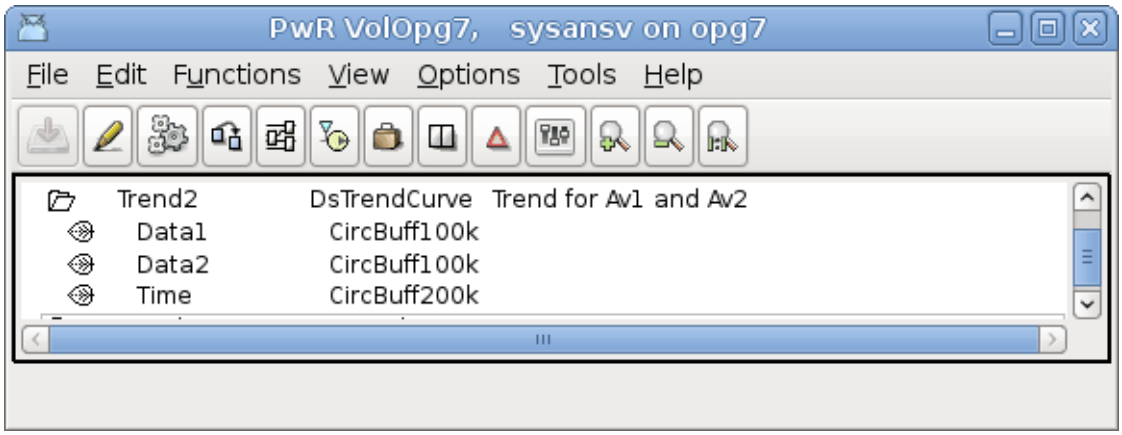
Custom logotype

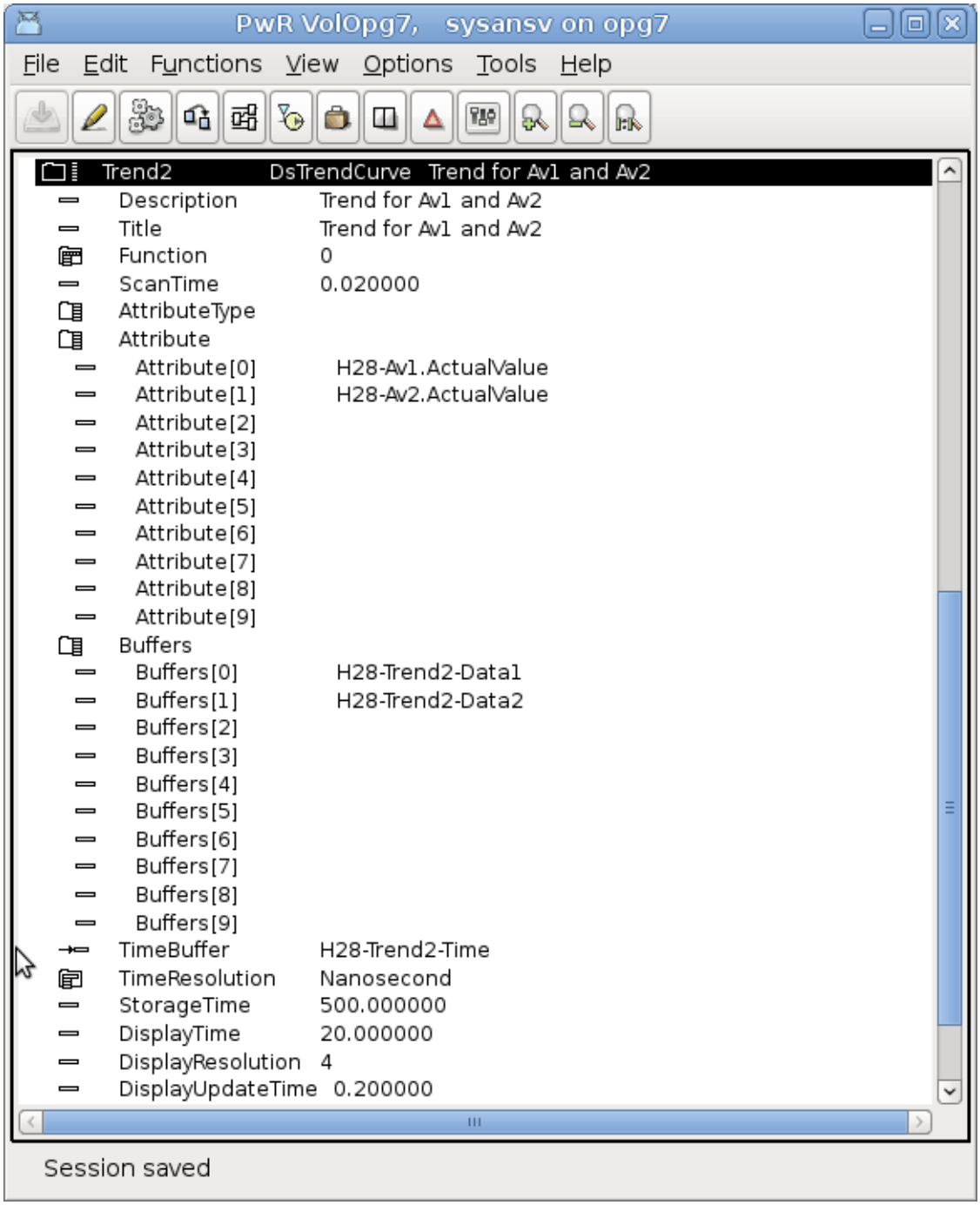
The Proview icon it the operator window can be replace by a custom logotype by placing an png-image in $pwrp_exe/pwrp_logotype.png. See the SSAB logo below.



Trend curves

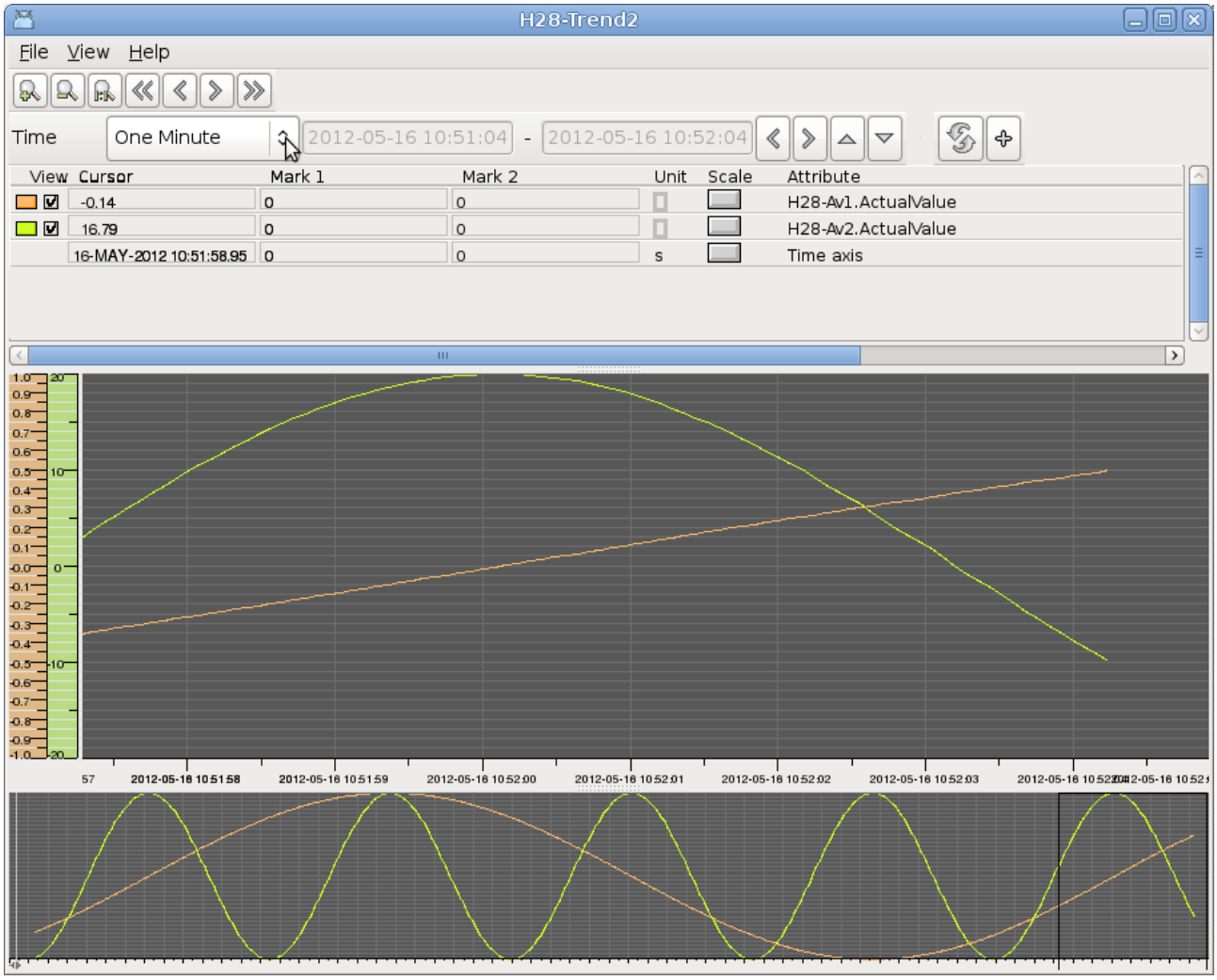
A new trend object, DsTrendCurve, is implemented. The old trend object where limited to ~500 sampes and a scan time of 1 s. The new object are configured with external buffer objects where the size of these buffer objects sets the limitation of the number of samples. A CircBuff100k as in the example below can contain 25000 samples. Also the speed is increased with scan times down to 20 ms. A snapshot function is added where a snapshot of the current state is stored and viewed in a curve window where it can be analysed in more detail. The snapshots can be stored to file and opened again at a later occasion.


PwR VolOpg7, sysansv on opg7
File Edit Functions View Options Tools Help
Trend2 DsTrendCurve Trend for Av1 and Av2
Data1 CircBuff100k
Data2 CircBuff100k
Time CircBuff200k


PwR VolOpg7, sysansv on opg7
File Edit Functions View Options Tools Help
Trend2 DsTrendCurve Trend for Av1 and Av2
Description Trend for Av1 and Av2
Title Trend for Av1 and Av2
Function 0
ScanTime 0.020000
AttributeType
Attribute
Attribute[0] H28-Av1.ActualValue
Attribute[1] H28-Av2.ActualValue
Attribute[2]
Attribute[3]
Attribute[4]
Attribute[5]
Attribute[6]
Attribute[7]
Attribute[8]
Attribute[9]
Buffers
Buffers[0] H28-Trend2-Data1
Buffers[1] H28-Trend2-Data2
Buffers[2]
Buffers[3]
Buffers[4]
Buffers[5]
Buffers[6]
Buffers[7]
Buffers[8]
Buffers[9]
TimeBuffer H28-Trend2-Time
TimeResolution Nanosecond
StorageTime 500.000000
DisplayTime 20.000000
DisplayResolution 4
DisplayUpdateTime 0.200000
Session saved

Configuration

The trend is configured by a DsTrendCurve object and 10 different attributes can be stated in the Attribute array. For each attribute a storage buffer of type CircBuff has to be created and stated in the Buffers attribute. The size of the buffer object has to correspond to the number of samples. Also a time buffer can be stated, this is only needed if the snapshot function is used. The time resolution determins the size of the time buffer. For a resolution of 1 second, 4 bytes per sample is used, and for resolution Nanoseconds 8 bytes is used.



Plc compile with warnings

Compilation of the plc code is done with the Wall switch. This will affect the code in CArithm and DataArithm that will be exposed of a harder syntax control.

Name changed of option file and plc exectuable

The name of the option file for the plc is changed. The previous name

`$pwrp_exe/plc_'nodename'_'busid'.opt`

should be changed to

`$pwrp_exe/plc_'nodename'_'busid'_'name'.opt`

where 'name' is the name of the PlcProcess object in lower case (default name is 'plc'), eg

```
$pwrp_exe/plc_aristotle_0999_plc.opt
```

Also the name of the plc executable is changed. Prevously a plc version was included in the name. This is now removed and replaced by the name of the PlcProcess object instead, eg

```
$pwrp_exe/plc_aristotle_0999_plc
```

Multiple plc processes with CPU affinity

The plc can divided into several plc processes. Each plc process is configured with a PlcProcess object. If is possible to set the CPU affinity to a process by stating an suitable CPU mask in the CpuMask attribute of the PlcProcess object.

V4.8.6 Additions

Pie chart and bar chart in java

Pie chart and bar chart was introduced in V4.8.5. Now it also possible to export the pie and bar charts to java.

Java Trend and Fast curves

Trend curves and Fast curves can be displayed in the java and web interface. The curves are opened from the Trend or Fast methods in the object popup menu, or with the commands 'open trend' or 'open fast'.

Class editor function object copy

Previously the symbolic references with $PlcMain, $PlcFo and $IoConnect was lost when a component class with a function object was copied. Now these references are kept and translated to the new class identity. Note that all the component classes has to be copied together, otherwise the symbolic references can not be translated.

Plc object DataArithmL added

The DataArithmL is similar to DatArithm, but has an increased code buffer. The maximum number of code characters is a DataArithm is 1023, while a DataArithmL can contain 8191 characters.

Xtt operator log configured in OpPlace object

The operator log can be started by setting the OperatorLog bit in the Options attribute of the OpPlace object.

Disable Qcom links between all nodes in a project

By default, all nodes configured in a project in the same bus, will create a Qcom and nethandler link between them. Now it is possible to disable this auto connect function, and the links can be configured individually for each node with FriendNodeConfig objects. The auto connect function is disabled by setting QcomAutoConnectDisable to No in the BusConfig object in the directory

volume.

Display values in Xtt as hex, octal and binary

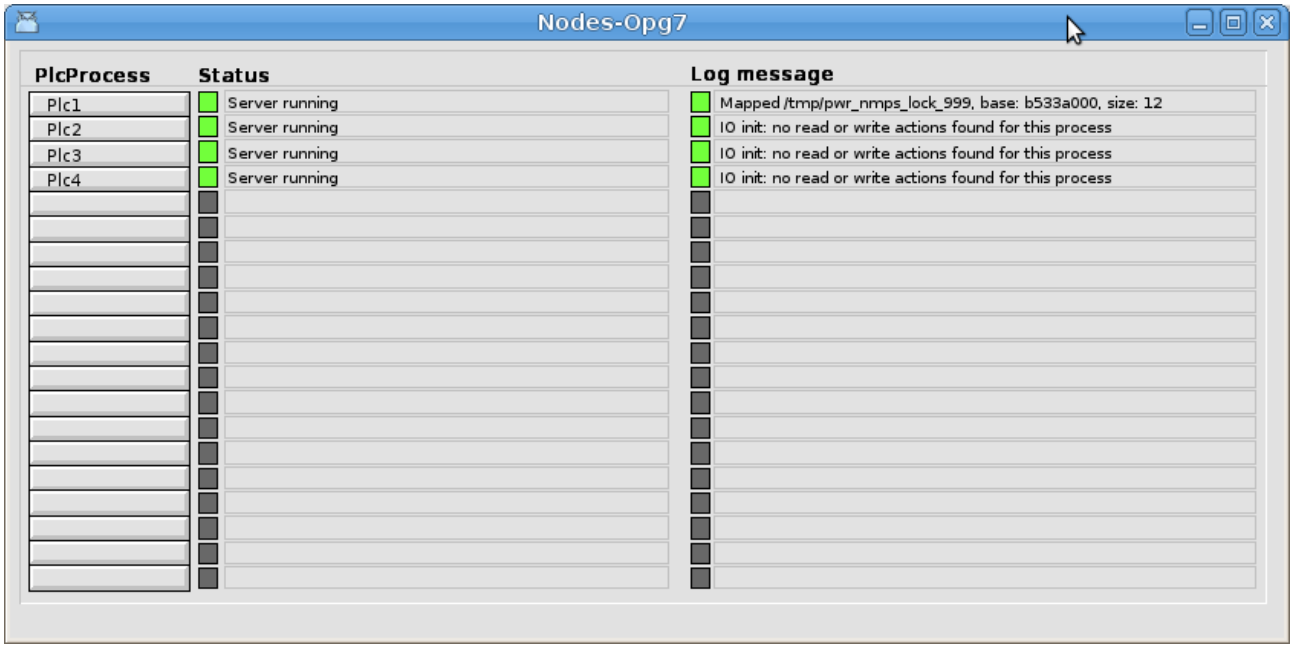
A value displayed in the runtime navigator can be displayed with another format than the default with the command 'set display'. The command will affect the conversion of the select attribute. Valid commands are

```
xtt> set display hexadecimal
xtt> set display hexadecimal
xtt> set display decimal
xtt> set display octal
xtt> set display binary
xtt> set display integer
xtt> set display float
xtt> set display identity
xtt> set display default
```

The 'set display identity' is used for object identities and attribute references and will show the identity value, and not convert to object or attribute name.

Several plc processes in System Status graph

Support for several plc processes was introduced in V4.8.5. Now the status of the plc processes are shown in the System Status graph. When clicking on the plc button, a list of the plc processes are viewed, displaying current status and log message. When clicking on a plc process, the ordinare plc process graph for this process is opened.

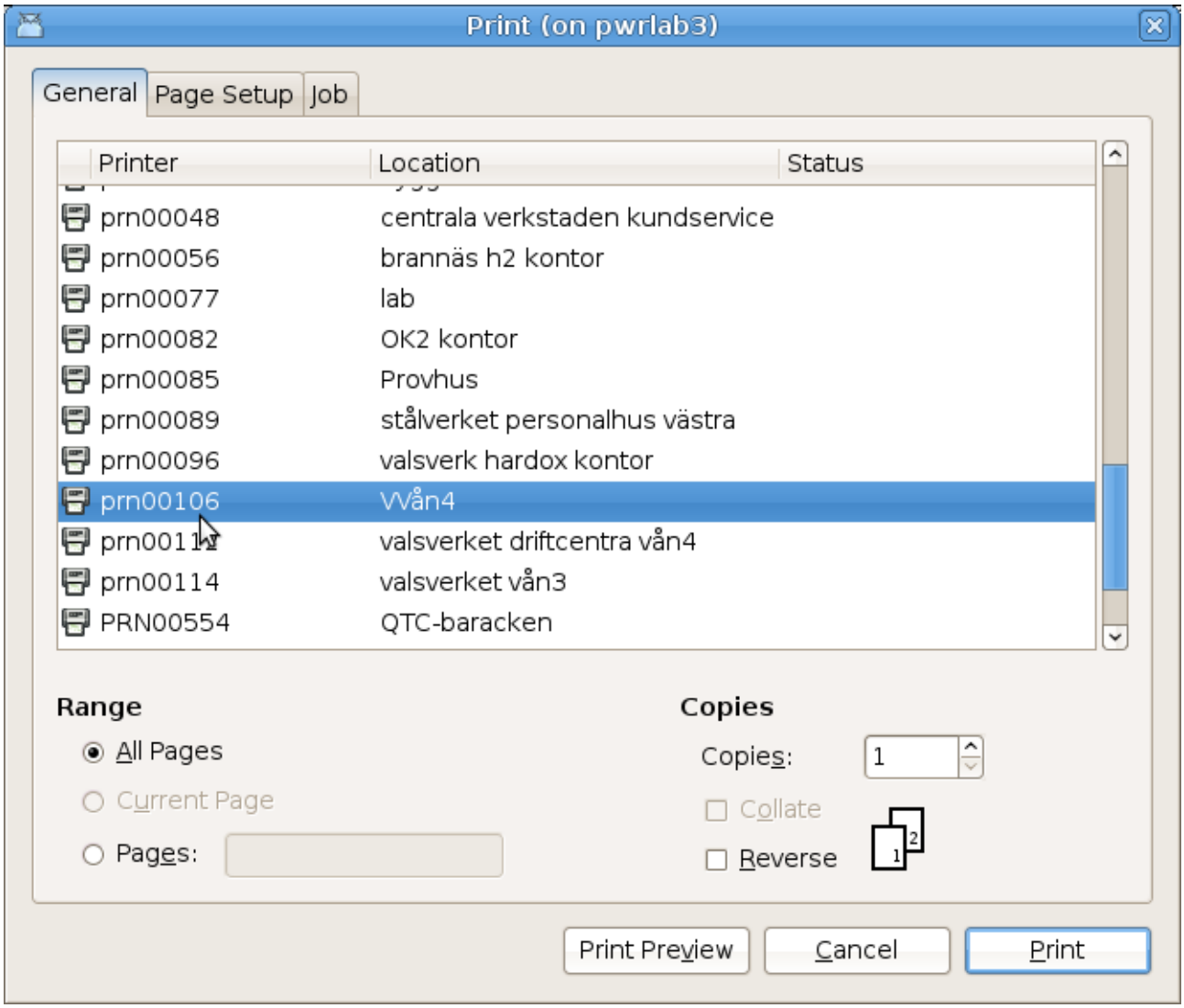


Also soft restart is supported for several plc processes in V4.8.6. The restart sequence will count in start signals from all processes.

Print dialog

The standnard gtk print dialog is added to most windows in the development and operator

environment, not yet to Ge graph though. In the operator environment the print dialog is disabled by default, as other windows are locked for input events when the dialog is opened. Enable the dialog in the Options attribute in the OpPlace object. When the dialog is disabled, it is possible to state a default printer queue in the Printer attribute of the OpPlace object. In this case the print is sent directly to the printer, without opening the print dialog.



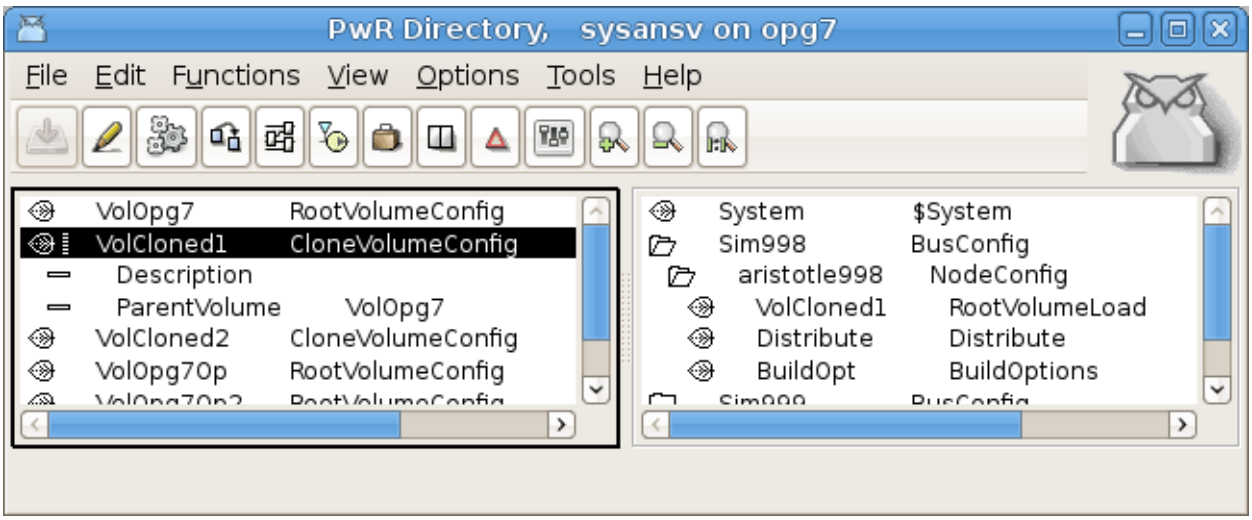
Plc trace version control

A version check between the version of the plc window in the database and the trace-file is added. If a version mismatch is detected when opening trace for a plc window, an error dialog is displayed.

Volume clones

When you have two identical plants, that is controlled by two more or less identical control systems, it's now possible to have one single database for one the the plants, and to generate a slightly modifed clone for the other plant.

A clone volume is configured with a CloneVolumeConfig in the directory volume. In the ParentVolume attriubute the parent volume of the clone is stated. Then a NodeConfig for the node of the clone is created with a RootVolumeLoad with the same name as the cloned volume.



A clone is generated by opening the parent volume and activating Functions/Clone Volume in the configurator. Now a temporary copy of the parent volume is created with the volume identity and name of the clone volume. A script, $pwrp_cnf/'volumename'_clone.pwr_com, is exectuted, and in this script is is possible to insert statments to change object names, set attribute values etc. Then the plc windows are redraw and flow-files are generated. The PlcPgm's are compiled, dbs-files and crossreference files are generated. Now the volume is cloned, and the node of the clone volume can be built and distributed.

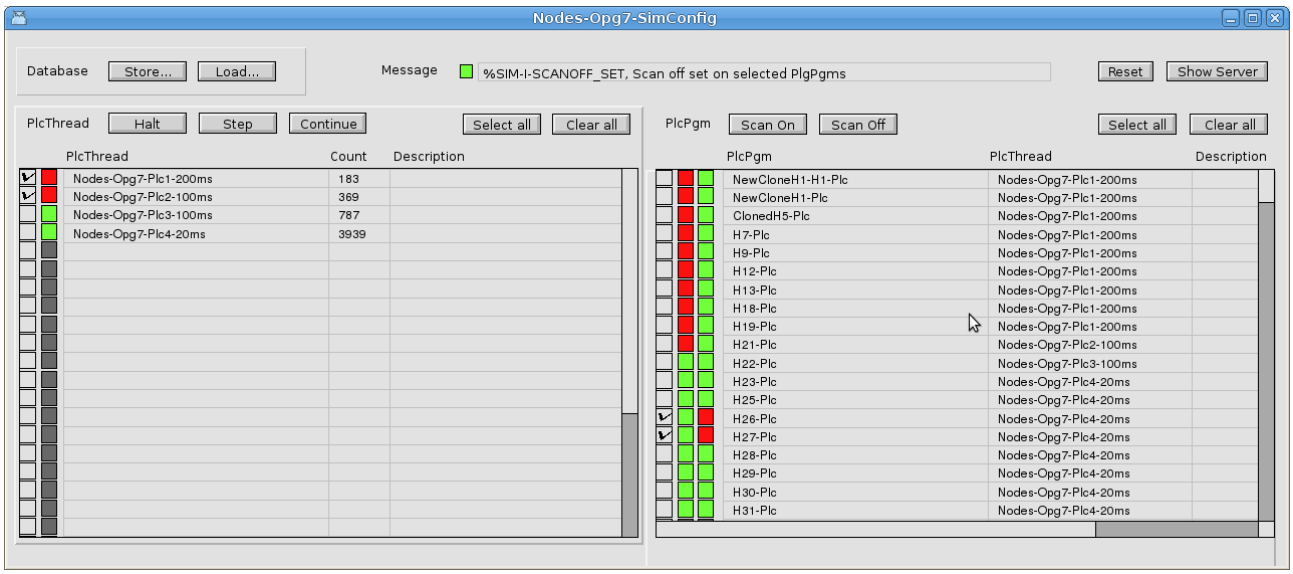
Here is an example of a modification script for VolCloned1, $pwrp_cnf/volcloned1_clone.pwr_com

```
main()

  rename /source=Op7 /name=Op8
  rename /source=Nodes-Op7 /name=Op8

endmain
```

Simulate server

A simuate server to increase the simulate functionallity is added in 4.8.6. The server is configured with a SimulateConfig object in the node hierarchy. When this object is present, the simulate server, rt_sim, is started. From the object graph of the SimulateConfig object it is possible to

- Start and stop plc threads individually.
- Step one scan in stopped threads.
- Enable and disable execution of PlcPgm's (Scan On and Scan Off).
- Take a snapshot of the current database and store to file. This requires that all threads are halted.
- Load a stored file into the database.



Sev database statistic table added

A table, sev_stat, is added to the history database to show the current load and some statistics of the server.

The table show the

1. The current load, ie database execution time in percentage.
2. Medium load.
3. Number of data items stored per second, current value.
4. Number of data items stored per second, medium value.
5. Number of data storage messages.
6. Number of data get messages.
7. Number of items messages.
8. Number of event storage messages.

```
mysql> select * from sev_stat;
| 7.24149 | 7.90368 | 907.713 | 908.8 | 32321 | 0 | 1 | 0 |
1 row in set (0.00 sec)
```

Sev database with SQLite

In addtion to MySQL, SQLite can be choosen as database in the sev server. The choice is made in the SevServer object, if Proview runtime is started, or in the /etc/proview.cnf file with the variable 'sevDatabaseType' which can have the value 'sqlite' or 'mysql'.

```
sevDatabaseType sqlite
```

Ge Axis dynamics

A new type of dynamics for axis object is added in Ge. The min and max value of the axis can be connected to attribute is the database, and the scale is updated when a min or max values is changed. New axis objects with this dynamic is found in the Analog map in the Ge editor.

Upgrade from V4.8.X to V4.8.6

Enter the administrator and change the version of the project to V4.8.6. Save and close the administrator.

I you have any class volumes, enter the class editor and build the volume.

Enter the configurator for each root volume and activate 'Function/Update Classes' and build.

Note ! The update procedure below doesn't have to be executed in this case.

Upgrade procedure

The upgrading has to be done from any version in the interval V4.7.0. 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->V4.5.0->V4.6.0->V4.7.0->V4.8.6

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 `sdf` 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
    dirvolume    Load directory volume
    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 renamedb dirvolume loaddb 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 create for the directory volume and one for every other rootvolume

```
> cd $pwrp_db
> ls -l *.wb_dmp
-rw-rw-r-- 1 cs pwrp 1771 2010-03-26 16:32 directory.wb_dmp
-rw-rw-r-- 1 cs pwrp 7467 2010-03-26 16:32 volopg2.wb_dmp
```

Linux release upgrade

If you are using Ubuntu 9.4 or Fedora 10 you need to upgrade the linux release and install the pwr48 package.

Change version

Enter the administrator and change the version of the project to V4.8.0. 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 you have to

answere whether to continue with the next pass or not.
Start the script with

```
> upgrade.sh
```

Start from the classvolumes pass.

```
Enter start pass [classvolumes] >
```

classvolumes

Create loadfiles and structfiles for the class volumes.

renamedb

Store the old databases under the name `$pwrp_db/'volumename'.db.1.`

cnvdump

Converts values of Profibus module objects.

loaddb

Create databases and load the dumpfiles into them.

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/x4-7-1/rls
bash: cd: /data0/pwrp/opg2/src/login: No such file or directory
>
> upgrade.sh

  upgrade.sh  Upgrade from V4.7.0 to V4.8.0
```

```
  Pass

    classvolumes   Create loadfiles for classvolumes.
    renamedb       Rename old databases.
    loaddb         Load dumpfiles.
    Cnvdump        Convert the dumpfiles.
    compile        Compile all plcprograms in the database
    createload     Create new loadfiles.
    createboot     Create bootfiles for all nodes in the project.


-- Upgrade opg2

Enter start pass [classvolumes] >

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

--------------------------------------------------------------------------
Pass cnvdump
--------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
/data0/pwrp/opg4/src/db/volopg2.wb_dmp

--------------------------------------------------------------------------
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 VolOpg
I Volume VolOpg 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 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 -O3 F1-Z1-Plc-W
-- Plc plcpgm compiled for x86_linux optimized -O3 F1-Z1-Plc
-- Plc window generated            F1-Z2-Plc-W
-- Plc window compiled for x86_linux optimized -O3 F1-Z2-Plc-W
-- Plc plcpgm compiled for x86_linux optimized -O3 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 'VolOpg'...
-- Open file...
-- Successfully created load file for volume 'VolOpg'
-- 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

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

Proview 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 -O3 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
>
>
```

Appendix A Embedded Linux

Proview is adapted to be built for embedded Linux systems with cross compilation in a Linux host environment.

First the Proview runtime module of the base system has to be built, and then a project is created where also the plc executable is built with cross compilation. The following example will describe a build for the ARM architecture with the cross compilation tools arm-linux-gnueabi-gcc, arm-linux-gnueabi-g++ and arm-linux-gnueabi-ar.

The runtime module build is dependent on an development installation or a complete build on the host system. Platform independent files as loadfiles and java archives are copied from the host release to the embedded build tree, also build tools in the host release are used to perform the build.

Environment variables defining the cross compilation tools, and the path to the exe directory of the host release has to be defined before starting the build.

```
export pwre_cc=arm-linux-gnueabi-gcc
export pwre_cxx=arm-linux-gnueabi-g++
export pwre_ar=arm-linux-gnueabi-ar
export pwre_host_exe=/usr/pwr47/os_linux/hw_x86/exp/exe
```

The tool to build Proview from sources, pwre, also has to be initialized

```
export pwre_env_db=~/pwre_env_db
export pwre_bin=~/pwrsrc_4.7.1/src/tools/pwre/src/os_linux
source $pwre_bin/pwre_function
```

Follow the Build from sources guide to build from the source code with the following modifications.

When adding the pwre environment, state the import root to the hw directory of the host release

```
Import root:  /usr/pwr47/os_linux/hw_x86
```
and set hardware to arm.

```
Hardware: arm
```

```
> pwre add armx471
 Source root [ ]? /home/pwrd/pwrsrc_4.7.1/src
 Import root [ ]? /usr/pwr47/os_linux/hw_x86
 Build root  [ ]? /home/pwrd/pwrrls_4.7.1
 Build type  [dbg]?
 OS          [linux]?
 Hardware    [ ]? arm
 Description [ ]? X4.7.1 for ARM
```

Initialize the arm environment

```
> pwre init armx471
```
Create the build tree

```
> pwre create_all_modules
```
Import files from the import release

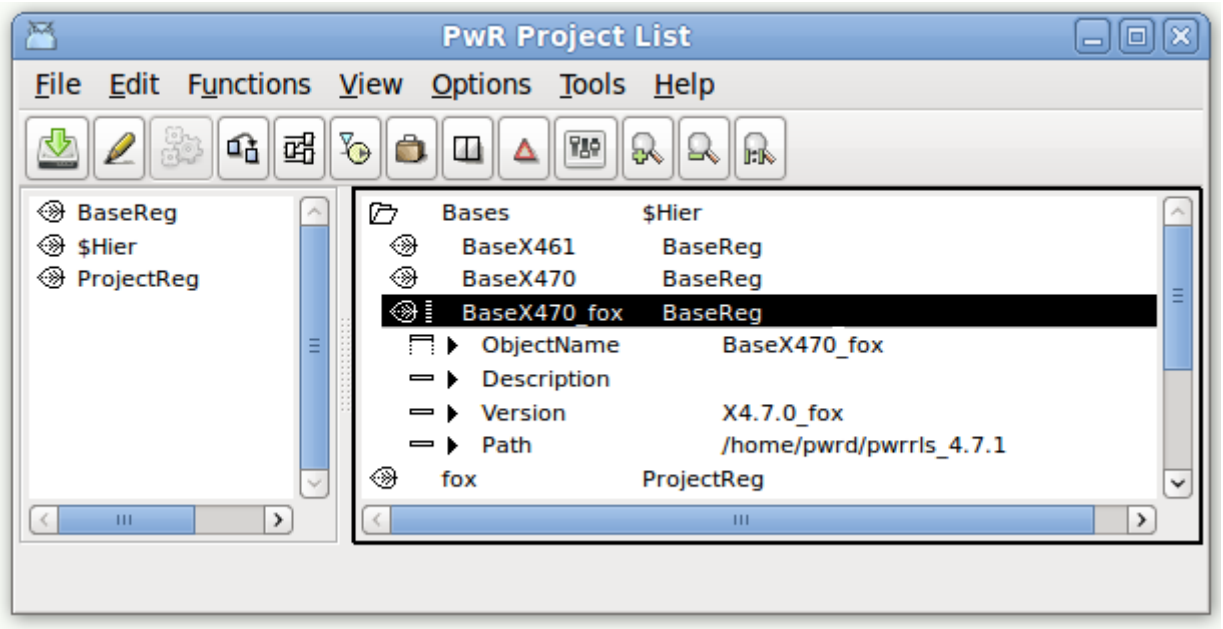
```
> pwre import rt
```
If the java archives are to be a part of the release these can be imported with the command

```
> pwre import java
```
Build the runtime module. If you want to customize the build you can choose what additional modules you want to build in the file $pwre_bin/ebuild.dat

```
> pwre ebuild rt
```

When the build is performed, create the /usr/pwrrt/exe, /usr/pwrrt/load directories in the embedded file system and copy the rt_ files to the exe directory, and .dbs -files to load directory.

Define the embedded release in the project list with a BaseReg object and insert to path to the release.



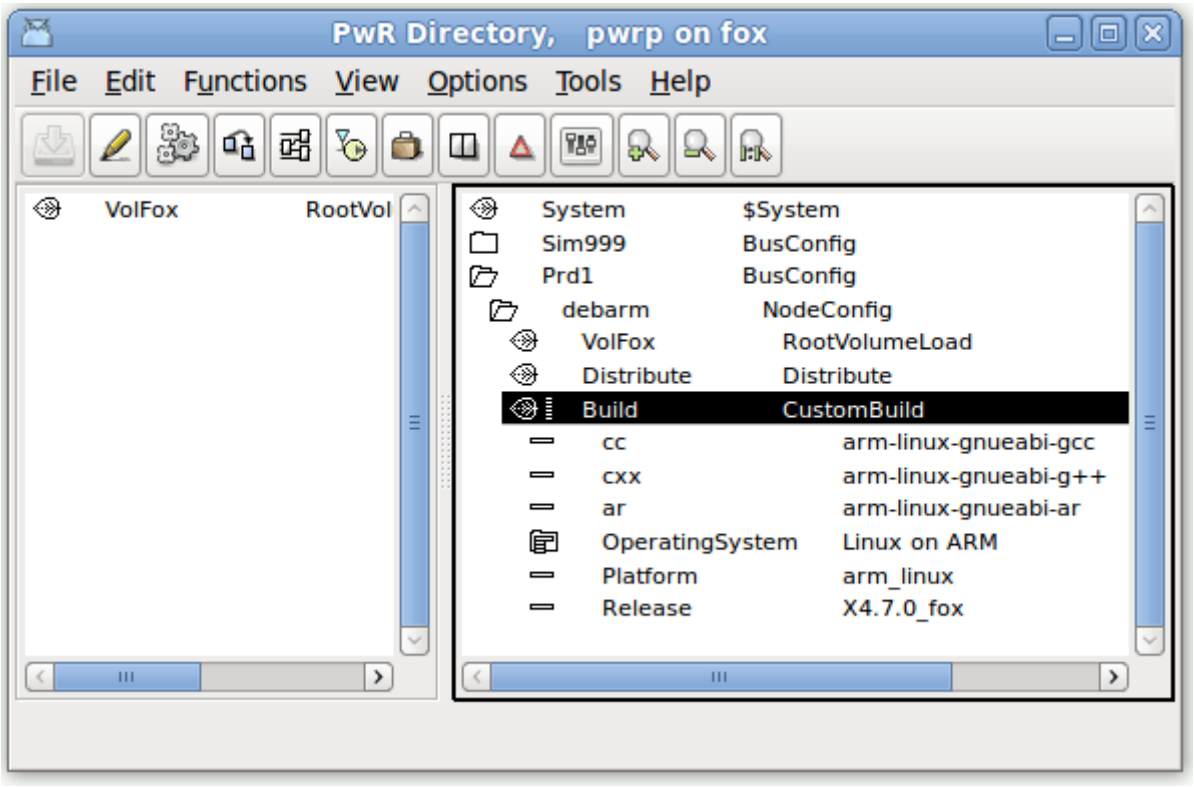
Project

The project is created and developed and simulated in the host release environment.

Set the OperatingSystem of the NodeConfig object for the embedded node in the directory volume to CustomBuild.

Create CustomBuild object below the NodeConfig object, that points out the embedded release and platform, and that defines the embedded toolchain.

The CustomBuild object will create a script, $pwp_exe/custom_build.sh that sets up the embedded build environment. If you need to make additional changes you can remove CustomBuild object and insert the changes in custom_build.sh.



Installing Proview

The following directories should be created in the root files system and the following files should be copied to them.

Proview runtime

`/etc`
Copied from $pwre_croot/src/tools/bld/pkg/deb/pwrrt

```
pwrp_profile
proview.cnf
```

`/usr/pwrp/adm/db/`
Copied from $pwra_db

```
pwr_user2.dat
```

`/usr/pwrrt/exe`
Copied from $pwr_exe (/home/pwrd/pwrrls_4.7.1/os_linux/hw_arm/rt/exe)

```
pwr_pkg.sh
pwr_stop.sh
rs_remote_3964r
rs_remote_logg
rs_remote_modbus
rs_remote_serial
```

```
rs_remote_tcpip
rs_remotehandler
rt_alimserver
rt_bck
rt_emon
rt_fast
rt_ini
rt_neth
rt_neth_acp
rt_print.sh
rt_prio
rt_qmon
rt_rtt
rt_sevhistmon
rt_statussrv
rt_sysmon
rt_tmon
rt_trend
rt_webmon.sh
rt_webmonelog.sh
rt_webmonmh.sh
```

/usr/pwrrt/load

Copied from $pwr_load

```
abb.dbs
basecomponent.dbs
inor.dbs
klocknermoeller.dbs
nmps.dbs
opc.dbs
otherio.dbs
othermanufacturer.dbs
profibus.dbs
pwrb.dbs
pwrs.dbs
remote.dbs
rt.dbs
siemens.dbs
ssabox.dbs
telemecanique.dbs
tlog.dbs
wb.dbs
```

/usr/pwrrt/lib

Copied from $pwr_lib

```
pwr_beans.jar
pwr_jop.jar
pwr_jopc.jar
pwr_rt.jar
pwr_rt_client.jar
```

Project

/pwrp/common/load
Copy from $pwrp_load

```
Loadfile ('rootvolumename'.dbs)
ld_node file (ld_node_'nodename'_'busid'.dat)
ld_boot file (ld_boot_'nodename'_'busid'.dat)
ld_appl file (ld_appl_'nodename'_'busid'.txt)
flow-files (.flw)
crossreference files (rtt_crr*_'volumeid'.dat)
```

/pwrp/common/log
Create only this directory if you want a log-file for system messages. Note that this might wear out you flash memory.

/var/www
Copy from $pwrp_web

```
*.html
pwrp_'nodename'_web.jar
```

Copy from $pwr_lib

```
pwr_rt_client.jar
pwr_jop.jar
pwr_jopc.jar
```

/pwrp/arm_linux/exe
Copy from $pwrp_root/bld/arm_linux/exe

```
Plc executable (plc_'nodename'_'busid'_'version')
xtt_help.dat
```

Settings

Set the Qcom busid in /etc/proview.cnf, parameter qcomBusId.

Execute /etc/pwrp_profile in .bashrc for the root user

```
source /etc/pwrp_profile
```

Add startup-file for Proview in for example /etc/init.d. Copy from

```
$pwre_croot/src/tools/pkg/deb/pwrrt/pwr
```

Distribute

The distributor can be use to copy files to a running system. For single user system add the username to the bootnode in the NodeConfig object, e.g. root@mynode.

Appendix B Mac OS X

Build from source on Mac OS X 10.6 x86_64

Install Xcode from the installation CD, or download from
http://developer.apple.com/technologiew/xcode.html

Download fink from http://www.fink.project.org. Follow the instructions to install. Install for 64 bit (question in ./bootstrap).

Install gtk

> fink install gtk+2
> fink install gtk+2-dev

Install doxygen
> fink install doxygen

Download BerkeleyDB 4.8 from
http://www.oracle.com/technetwork/database/berkeleydb/download.
Build with
> cd ./build_unix
> ../dist/configure –enable-cxx
> make
> sudo make install

Download iconv from http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.13.1.tar.gz.
Build with ./configure, make, sudo make install.

Download Proview sourcecode and follow the Build from source guide.
To download with git, install git with
> fink install git

Before build:
> export PKG_CONFIG_PATH=/sw/fink/pkgconfig
> export PATH=$PATH:/sw/bin

Before you start Proview runtime:

Add to /etc/sysctl.conf

kern.sysv.shmmax=167772160
kern.sysv.shmseg=16
kern.sysv.shmall=65536