

Uppgradering Proview V4.0

Revision:
Version:

04 01 22
V4.0.1

Claes Sjöfors

Inledning	4
Nya funktioner.....	5
Heltals IO och aritmetik.....	5
Utvecklingsmiljön.....	5
Volymer och databaser.....	5
Klassvolymer.....	5
Subvolymer.....	5
Projektvolymer.....	6
localWb	6
BerkeleyDb.....	6
Projektadministratören	6
Wtt	6
Connect funktioner.....	6
Klass editor.....	6
Sammansatta funktionsobjekt	6
Distributören	7
Konfigurering	7
Paket.....	7
Pakethanterare	7
Distributörsfönstret.....	8
RunTime	9
Ge	9
Rounded RoundedRectangle	9
Dynamiken omarbetad.....	10
FillLevel.....	14
AnalogColor.....	16
ToolTip.....	16
Färgad text	16
300-färgers palette.....	17
3D	19
Kopplingar med kantlinjer och 3D.....	20
”Rörkopplingar”	22
Input focus.....	23
Menyer.....	24
Nya klasser.....	25
WbEnvironment.....	25
BusConfig	25
Distribute.....	25
SubVolume	25
SubVolumeConfig.....	25
SubVolumeLoad	25
IiArea	25
IoArea	25
Iv.....	25
Ii.....	25
Io.....	25
ChanIi.....	26
ChanIo	26
GetIv.....	26
GetIi.....	26
GetIo.....	26
StoIv	26
StoIi.....	26
StoIo.....	26

CStoIv	26
CStoIi	26
CStoIo	26
GetIp	26
StoIp	26
CStoIp	26
AtoI	27
ItoA	27
GetIGeneric	27
StoIGeneric	27
MaskToD	27
DtoMask	27
EnumToD	27
DtoEnum	27
GetDattr, GetAattr, GetSattr, GetIattr	27
StoDattr, SetDattr, ResDattr, StoDattr, StoAattr, StoIattr, StoSattr, CStoAattr, CStoIattr, CStoSattr	27
Föråldrade klasser	27
GraphDistribute	27
SystemDistribute	27
DbConfig	27
Övriga	28
Ändrade klasser	28
TemplateVärden	28
GetIp	28
StoIp	28
CStoIp	28
CArithm	28
DataArithm	28
Ai, Ao, Di, Do, Po, Co, ChanAi, ChanAit, ChanAo, ChanDi, ChanDo, ChanCo	28
Installation	28
Procedur för uppgradering	30
Ta en kopia av projektet	30
upgrade.sh	31
dumpdb	31
userclasses	32
dirvolume	32
cnvdirvolume	32
cnvdump	32
createvolumes	32
localwb	32
compile	32
createload	32
createboot	32
convertge	32
Övrigt	32

Inledning

Det här dokumentet beskriver nya funktioner i Proview V4.0, samt hur man uppgraderar från V3.4b och V3.9a.

Nya funktioner

Heltals IO och aritmetik

Det finns nu stöd för hantering av heltal i IO-hantering i form av signalerna Io (Integer output) and Ii (Integer input). Dessa kommer främst att användas för IO till profibusmoduler. Det finns även ett Iv objekt (Integer value). Io, Ii och Iv IO-kopieras precis som sina analoga och digitala motsvarigheter.

För användning av heltals signalerna i plc-programmet finns Get, Sto och CSto objekt. Dessutom har CArithm och DataArithm försäts med heltals ingångar och utgångar. Det finns även objekt för att konvertera analoga signaler till heltal och vv (AtoI resp ItoA). Objekten MaskToD och DToMask konverterar mellan bitmask och digitala signaler, EnumToD och DtoEnum konverterar mellan ett uppräkningsvärde och digitala signaler.

De GetIp, StoIp och CStoIp objekt som fanns i V3.4, som, förutom att lagra resp hämta ett heltalsvärde, även konverterade mellan heltal och analogt värde, har nu bytt namn till GetIpToA, StoAtoIp och CStoAtoIp. De nya GetIp, StoIp och CStoIp objekten konverterar inte till eller från analoga värden, däremot konverterar de mellan olika typer av integer typer. Ett integer attribut (Ip) kan var av typen Int32, UIn32, Int16, UInt16, Int8 eller UInt8, medan integer signaler och in och utgångar på plcobjekt alla är av typen Int32.

Utvecklingsmiljön

Volym och databaser

Det stora skillnaden ligger i konfigurationen av volymer och databaser.

Numera finns enbart en volym i varje databas, så begreppet databas identitet har försvunnit.

När man startar pwrns anger man nu volyms namn istället för databas identitet, t ex

➤ pwrns volvwxn1t

Man kan bara editera en volym åt gången, övriga volymer kan man titta genom deras laddatafiler öppnas. En laddatafil innehåller nu all data om objekten som fanns i databasen. Tidigare innehöll den enbart data som behövdes i runtime. Det gör att man nu kan gå in i en laddatafil med navigatören och titta i objekt och plc-program, på samma sätt som man kan göra i den databas. Enda skillnaden är att det inte går att editera i en laddatafil.

Man kan även titta och kopiera från volymer i andra projekt genom att öppna laddafilerna från File/OpenFile i menyn i navigatören.

Även wb_load filer kan nu läsas in av navigatören. Detta utnyttjas bl a i den nya klass editorn.

Klassvolym

Klassvolymerna ligger aldrig i någon databas längre. Från wb_load filen genereras laddatafiler, som används både i utvecklingsmiljön och i runtime.

Subvolym

Tidigare har alla objekt på en nod tillhört en volym, rot-volymen. Nu kan man dela upp objekten på en nod i flera volymer. Objekten kan även ligga i subvolym som i runtime monteras av rotvolymen.

En subvolym konfigureras med objektet SubVolumeConfig i projektvolymen, på samma sätt som rot-volymen. Man måste även ange att subvolymen ska laddas i en nod genom att lägga ett SubVolumeLoad objekt under NodeConfig objektet. I rot-volymen ska det finnas ett MountObjekt objekt som monterar subvolymen.

Fördelen med subvolymen är att man enkelt kan flytta objekt mellan två noder, och att man delar upp objekten i flera databaser, vilket kan vara nödvändigt om man är flera som jobbar i ett projekt. Se BerkeleyDb nedan.

Projektvolymen

DbConfig objektet, med vilket man konfigurerade databaser, har utgått. Nu konfigurerar man volymerna med RootVolumeConfig och ClassVolumeConfig, på top-nivån i projektvolymens volymshierarki. Här kan man även lägga in ett WbEnvironment objekt, se nedan.

NodeConfig objektet låg förut på topnivå, men ska nu ligga under ett BusConfig objekt. Ett BusConfig objekt konfigurerar en QCOM bus, och NodeConfig objekten för de noder som tillhör bussen ska ligga under BusConfig objektet.

GraphDistribute och SystemDistribute objekt har utgått, och ersatts av Distribute och ApplDistribute.

localWb

Lokala listdescriptor objekt och lokal templateobjekt ligger i volymen localWb. localWb läses in när utvecklingsmiljön startar från filen \$pwrp_db/wb.wb_load

BerkeleyDb

I 3.4 användes MySql som databas. Denna är nu ersatt av BerkeleyDb. En konsekvens av detta är att endast en session åt gången kan vara öppen mot en databas. Om flera måste jobba i samma volym, kan man lösa detta genom att dela upp den i subvolymen.

Projektadministratören

Från projektadministratören hanterar man numera även proview's användardatabas och registrering av volymer.

Wtt

Wtt har fått funktioner i menyn för att öppna laddatafiler och wb_load filer (File/Open File/Dbs... resp File/Open File/Wbl...).

Connect funktioner

Connect-funktionerna i plc-editorn och Ge-editorn, som aktiveras med Ctrl/DubbelKlick fungerar nu på utvalda objekt även i andra navigatörer. Dessutom får man automatiskt rätt copy-syntax med attribut och typ i Ge.

En ny funktion är att man kan sätta värdet på ett Objid attribut genom att välja ut ett lämpligt objekt, och klicka med Ctrl DubbelKlick MB1 på Objid attributet (i konfiguratören eller i attributeditorn).

Vissa objekt har fått nya Connect-metoder i popup menyn. Ett plcpgm kan kopplas till en tråd genom att man väljer ut PlcThread objektet och aktiverar ConnectThread i popupmenyn för PlcPgm objektet.

Klass editor

Det finns en klasseditor för att editera klass volymer.

En klassvolymkonfigureras i projektvolymen med ett ClassVolumeConfig objekt. Konfigureringen gör att en wb_load fil för klassvolymen att skapas: \$pwrp_db/'volymnamn'.wb_load. Denna kan som förut editeras för hand, men man kan även läsa in den i en klasseditor där man skapar klassdefinitions-objekten på samma sätt som i konfiguratören. I klassvolymen kan man även skapa funktions eller subrutins objekt för plc't.

Sammansatta funktionsobjekt

Sammansatta funktionsobjekt i plc't skapas på följande sätt. Man skapar en ny klass i klasseditorn och specificerar in och utgångar med \$Input och \$Output objekt. Under \$ClassDef objektet lägger man även ett PlcPgm objekt och

i detta editeras plc-koden för klassen. Ingångar läses med speciella objekt av typen GetDattr, GetAattr, GetIattr och GetSattr. Utgångar sätter man med objekten StoDattr, StoAattr, StoIattr, StoSattr, SetDattr, ResDattr, CStoAattr, CStoIattr och CStoSattr.

Man måste också skapa en lokal konfigurations fil för plc-paletten som innehåller den nya klassen.

Distributören

Distributören är helt omarbetat i V4.0. Den konfigureras med Distribute och ApplDistribute objekt (GraphDistribute och SystemDistribute har utgått), och packar ihop alla filer som ska distribueras till en nod i en zip-fil (ett paket). Zip-filen skickas över till målnoden och packas upp där av en paket-hanterare.

Konfigurering

Konfigureringen av distributören görs i projektvolymen, med Distribute och ApplDistribute objekt.

Distribute

Distribute objektet sköter distribution av standard-filer för en nod. Vissa filer, som kan vara specifika för en nod, hämtas i första hand från en filkatalog för noden \$pwrp_src/'nodnamn', och i andra hand från en gemensam katalog.

- Användar-databasen, \$pwra_db/pwr_user.dat eller nodspecifik katalog.
- Laddatafiler (dbs-filer, plc-programmet, rtt korsreferensfiler).
- Applikations fil, \$pwrp_load/ld_appl_'nodnamn'_'busid'.txt
- \$pwrp_load/pwrp_alias.dat
- Include-filer, \$pwrp_inc/*.
- Graph-filer, \$pwrp_exe/*.
- Xtt hjälptexter, \$pwrp_exe/xtt_help.dat eller nodspecifik katalog.
- Xtt resursfile (konfigurering av funktionstangenter), \$pwrp_pop/Rt_xtt eller nodspecifik katalog.
- Flow filer, \$pwrp_load/*.flw
- .rhosts fil, \$pwra_db/.rhosts eller nodspecifik katalog.
- Webb-filer, \$pwrp_web/*.jar, *.gif och *.html

Objektet läggs under NodeConfig objektet för aktuell nod.

ApplDistribute

Filer som inte hanteras av Distribute objektet, distribueras med ApplDistribute objekt, som lämpligen läggs som barn till Distribute-objektet.

Paket

Alla filer för en nod, som är specificerade med Distribute och ApplDistribute objekt, packas ihop i ett paket \$pwrp_load/pwrp_pkg_'nodnamn'_'version'.tgz. Paketet bör innehålla samtliga filer som krävs för att starta noden. Paketet innehåller dessutom en lista på alla filer, deras destination på målnoden och skapande-tid. Paketet kopieras med ftp till målnoden och läggs på /home/pwrp. Men hjälp av en pakethanterare (\$pwr_exe/pwr_pkg.sh) packas paketet upp och filerna flyttas ut till sina destinations-kataloger. Med pakethanteraren kan man även backa genom att packa upp äldre paket.

Pakethanterare

Pakethanteraren används för att packa upp paket, titta på innehållet i paket och avinstallera paket. Med 'pwr_pkg -l' listar man alla paket

```
$ pwr_pkg -l
pwrp_pkg_vhxnu4_0028.tgz
pwrp_pkg_vhxnu4_0029.tgz
pwrp_pkg_vhxnu4_0030.tgz
pwrp_pkg_vhxnu4_0031.tgz
```

'pwr_pkg -lp 'pkg'' visar även datum för paketet och med 'pwr_pkg -lf 'pkg'' kan man se vilka filer som ingår.

```
$ pwr_pkg -lp pwrp_pkg_vhxnu4_0029.tgz
pwrp_pkg_vhxnu4_0029.tgz vxhnu4 Version 29 19-DEC-2003 11:53:19.34
```

Om man vill backa till ett äldre paket görs detta med 'pwr_pkg -i 'pkg''

```
$ pwr_pkg -i pwrp_pkg_vhxnu4_0029.tgz
-- Removing package pwrp_pkg_vhxnu4_0031.tgz
-- Installing package pwrp_pkg_vhxnu4_0029.tgz
-- Unpack package pwrp_pkg_vhxnu4_0029.tgz
-- Move file to target directories
```

Distributörsfönstret

Distributörsfönstret är helt omarbetat, och hanterar nu paket istället för enstaka filer.

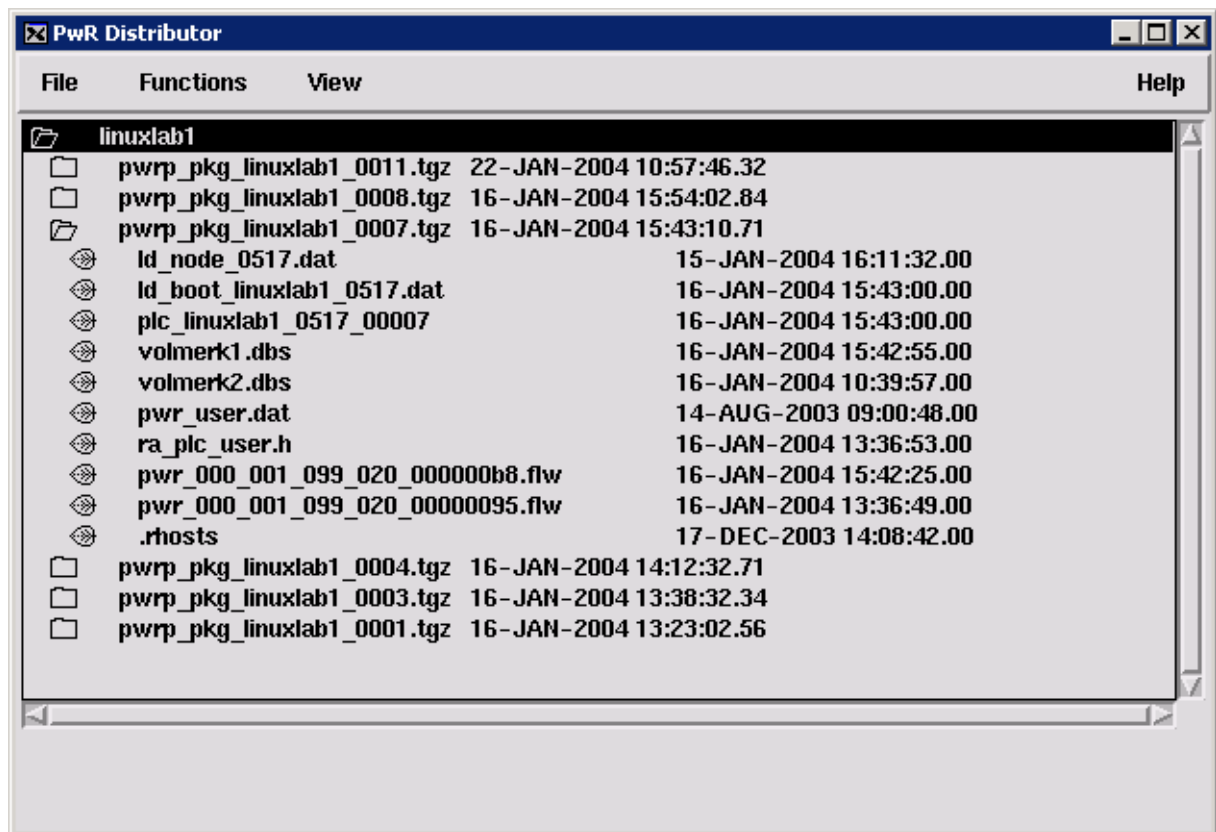
I Distributörsfönstret visas de noder som finns konfigurerade i projektet. För att skapa paket och distribuera dem väljer man ut en eller flera noder och aktiverar *Functions/Distribute* i menyn. För varje utvald nod skapas ett paket med samtliga filer som hör till noden. Paketet kopieras med ftp till målnoden och packas upp där.

Uppackningen sker via rsh, vilket kräver att .rhost filen innehåller den användare som utför distributionen. Man kan visserligen ange att .rhost filen ska finnas med i paketet, men vid den första distributionen måste .rhost filen skapas för hand.

Om man klickar på en nod kan man se tidigare skapade paket som hör till noden, och genom att klicka på ett paket kan man se de filer som paketet innehåller. Med *View/Display File Difference* i menyn kan man ange att endast filer som är ändrade sedan närmast tidigare paket visas.

Ett tidigare skapat paket kan distribueras genom att man väljer ut paketet och aktiverar *Functions/Distribute* i menyn.

Gamla paket kan rensas bort genom att man väljer ut dem och aktiverar *Functions/Delete Package* i menyn.



RunTime

Backup och Konsollog filer kan skrivas med environment variabler t ex "\$pwrp_log/console.log"

Skantiden på trace kan ändras i menyn (500 ms till 20 ms).

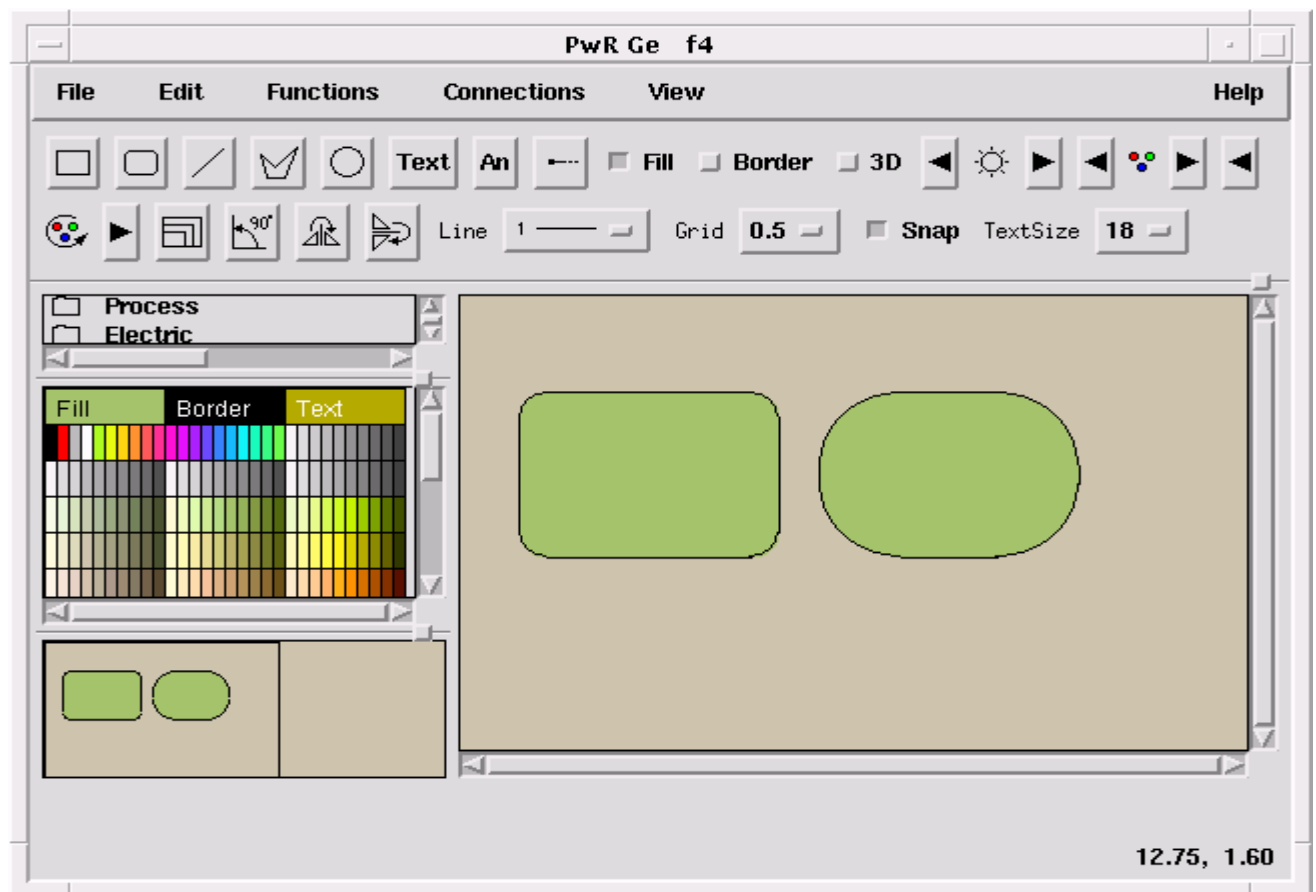
Ge

Ge har fått en rad nya funktioner, framför allt är dynamiken omarbetad.

Rounded RoundedRectangle

Basobjekten har utökats med en rektangel med rundade hörn.

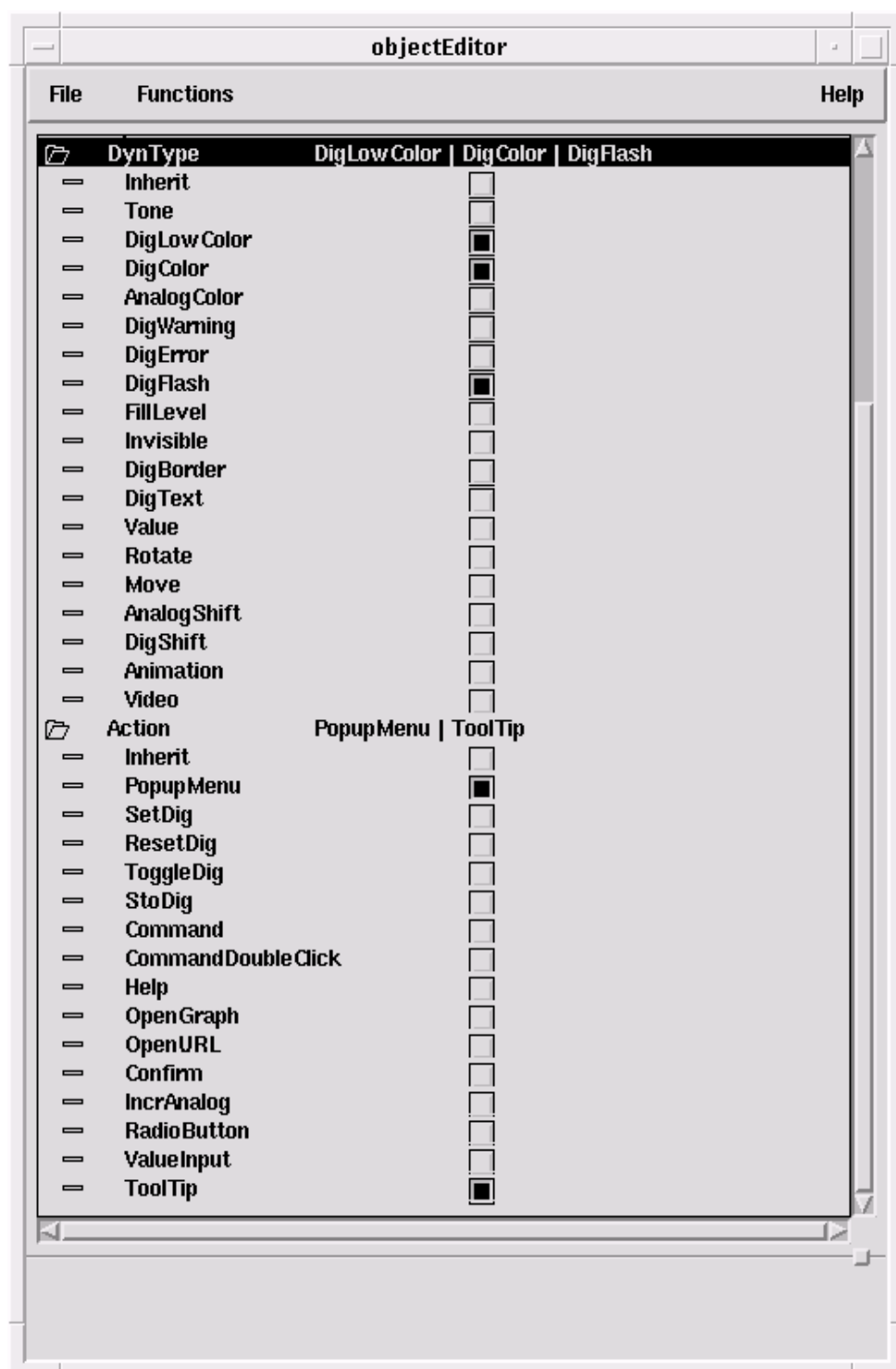
Fel!



Dynamiken omarbetad

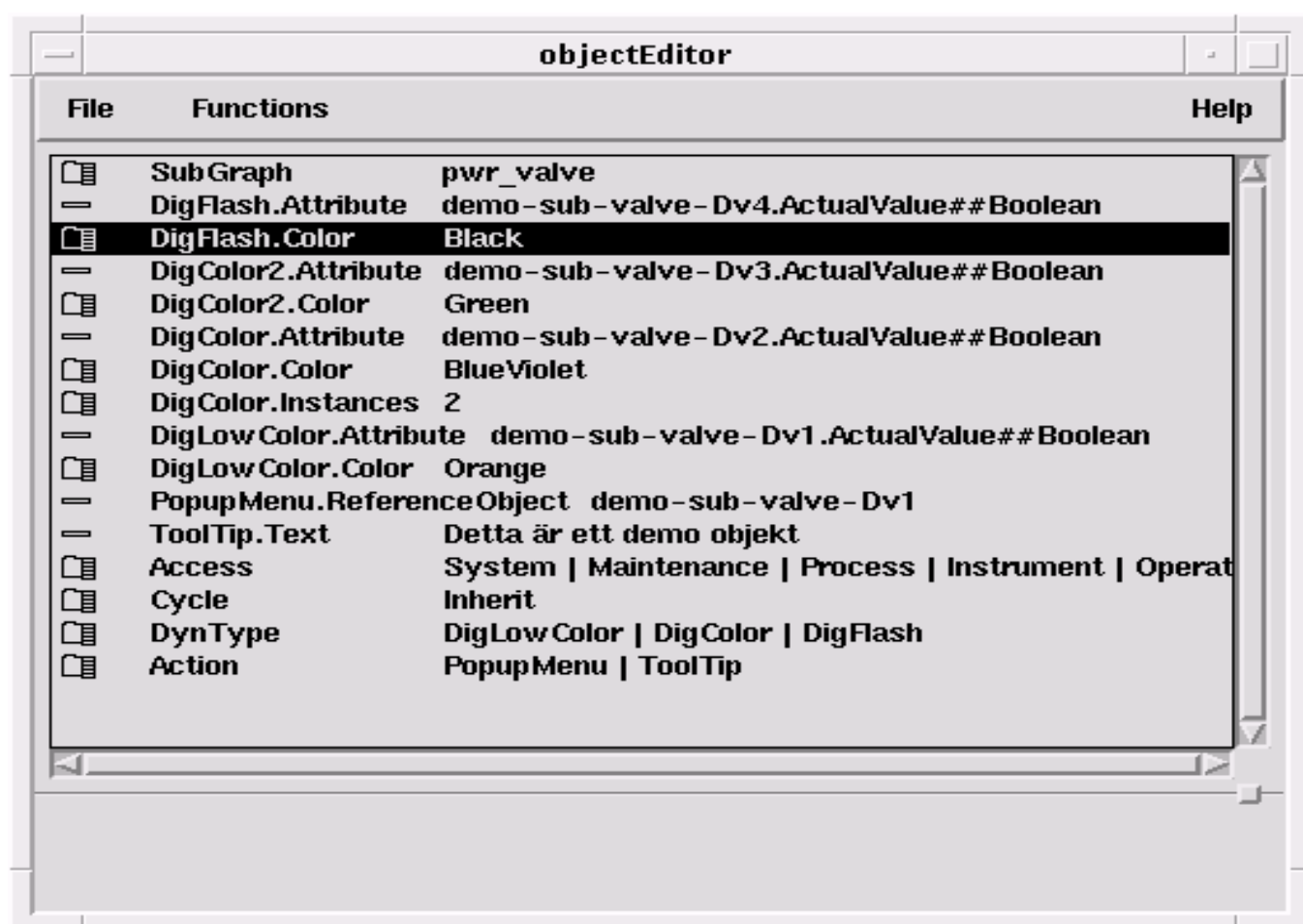
Det som i V3.4 betecknades som dynamik typ är nu uppdelad på två bitmaskar: dynamik och action. Med dynamik anges hur ett objekt ska påverkas av processen, t ex ändra färg, form, skalas eller flyttas. Med action anges vad som ska hända när man klickar på objektet, t ex visa en popupmenu, sätta en Dv. Den stora förändringen ligger i att olika typer av dynamik och action kan kombineras fritt för ett objekt. Nedan visas de olika typerna av dynamik och action.

Fel!



Objektseditorn är förändrad för att göra det möjligt att programmera alla olika kombinationer av dynamik och action. De egenskaper som hör till en viss dynamik eller action typ, markeras i attribut-namnet. Namnet för LowColor är nu DigLowColor.Color. För vissa typer av dynamik har begreppet instance införts. Dett innebär att en dynamik kan finnas i flera instanser för samma objekt. DigColor, som ändrar färg, beroende på värdet av en digital signal, kan finnas i flera instanser, vilket gör att man kan ha upp till 32 olika färger på ett objekt och koppla det till 32 digitala signaler.

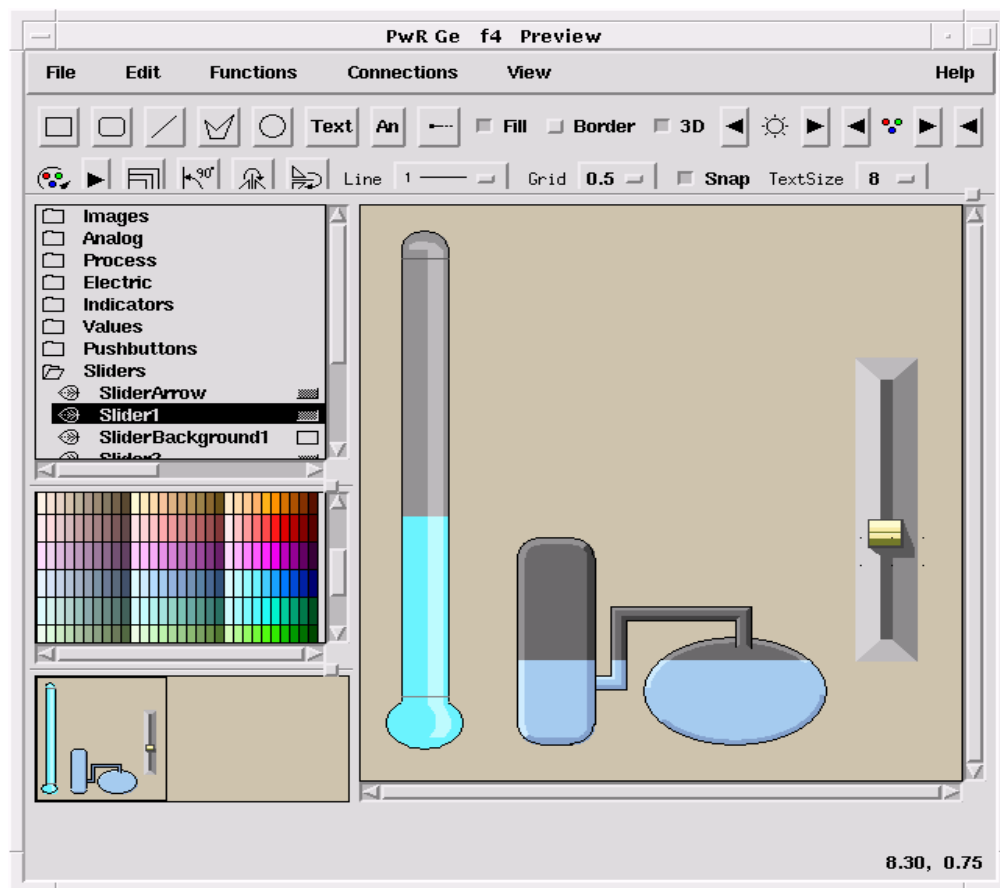
Fel!



FillLevel

FillLevel är en ny dynamik typ som färgar ett objekt till en viss nivå. Nivån kan ändras uppåt eller nedåt, åt vänster eller åt höger.

Fel!



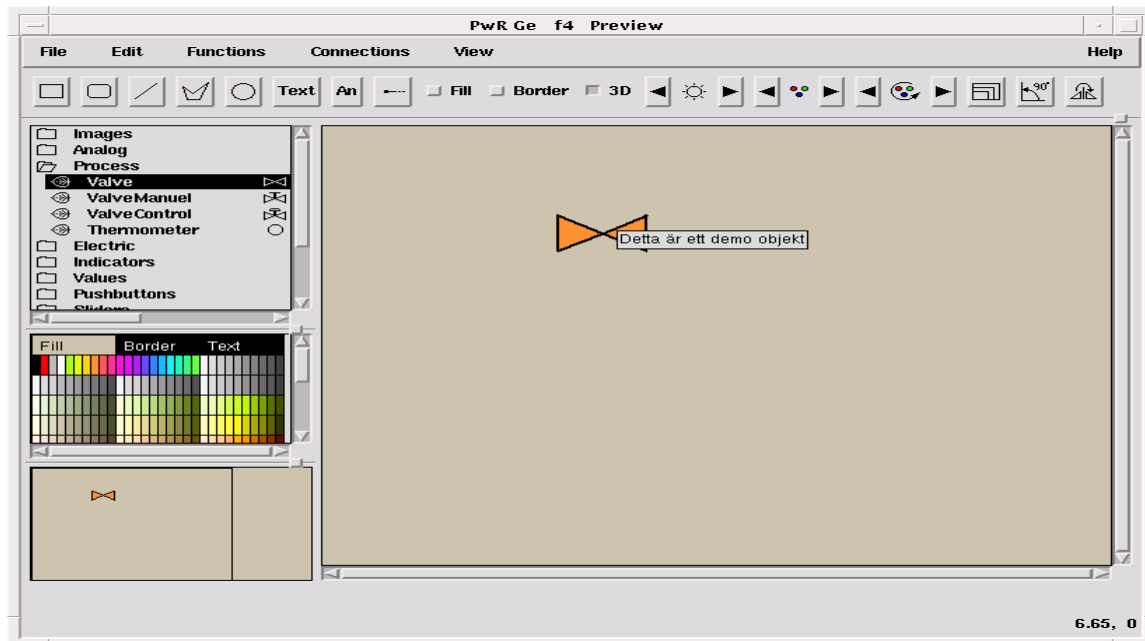
AnalogColor

AnalogColor kopplas till en analog signal, och ändrar färg på objektet när signalen passerar ett gränsvärde. Man kan ange om det är ett övre eller undre gränsvärde.

ToolTip

ToolTip visar en text om markören vilar på ett objekt.

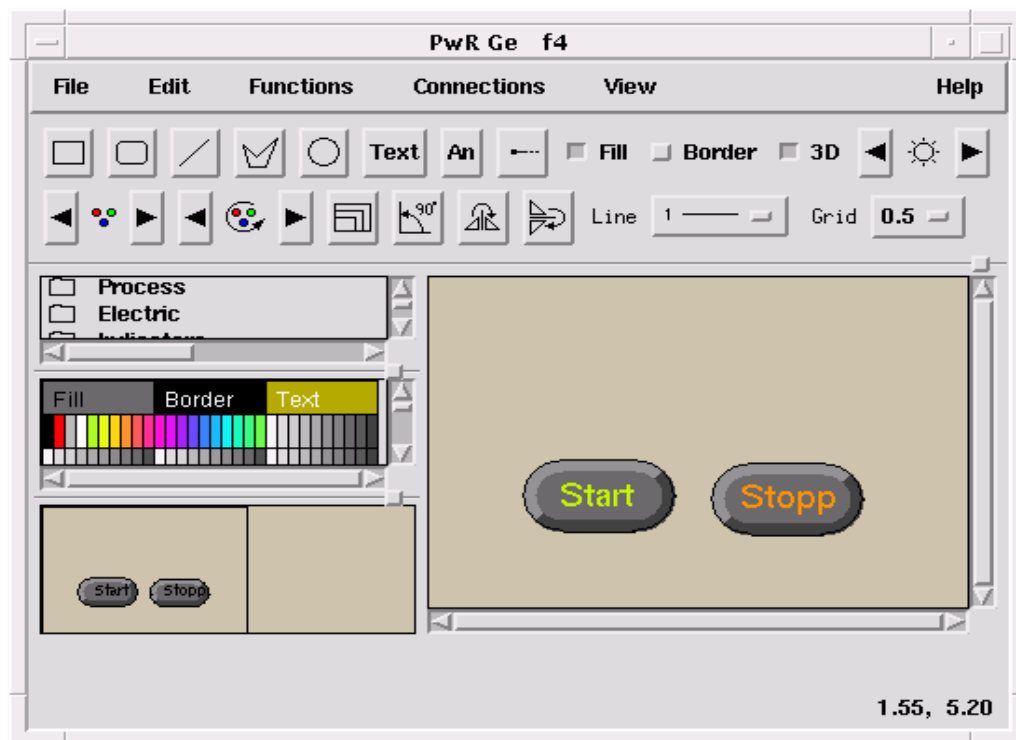
Fel!



Färgad text

Texter kan skrivas i olika färger. Textfärg väljs i paletten med Shift Click MB1

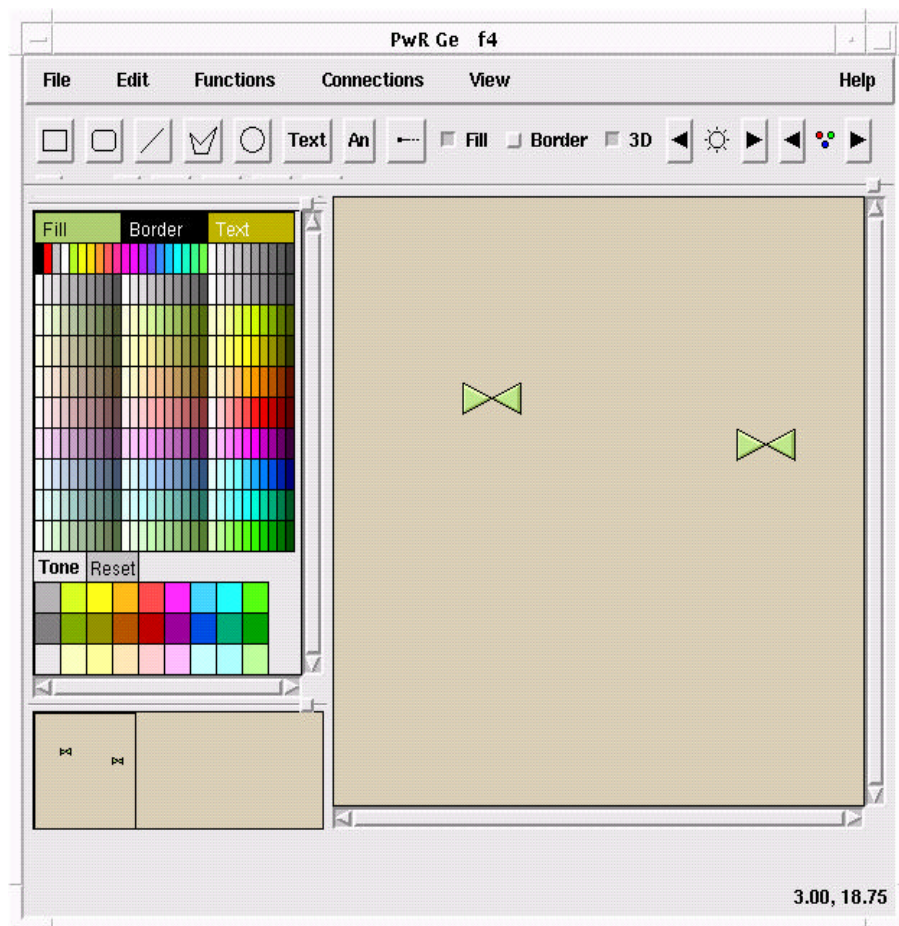
Fel!



300-färgers palette

Paletten är utökad till 300 färger (tidigare 100).

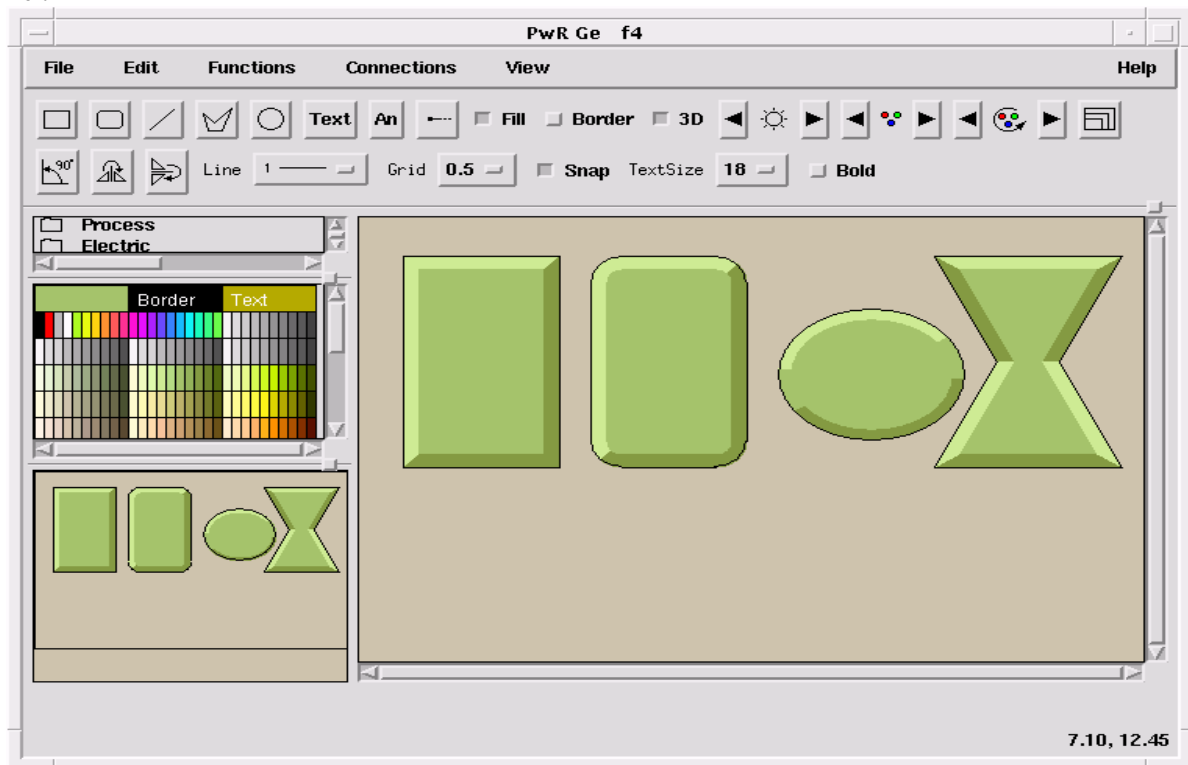
Fel!



3D

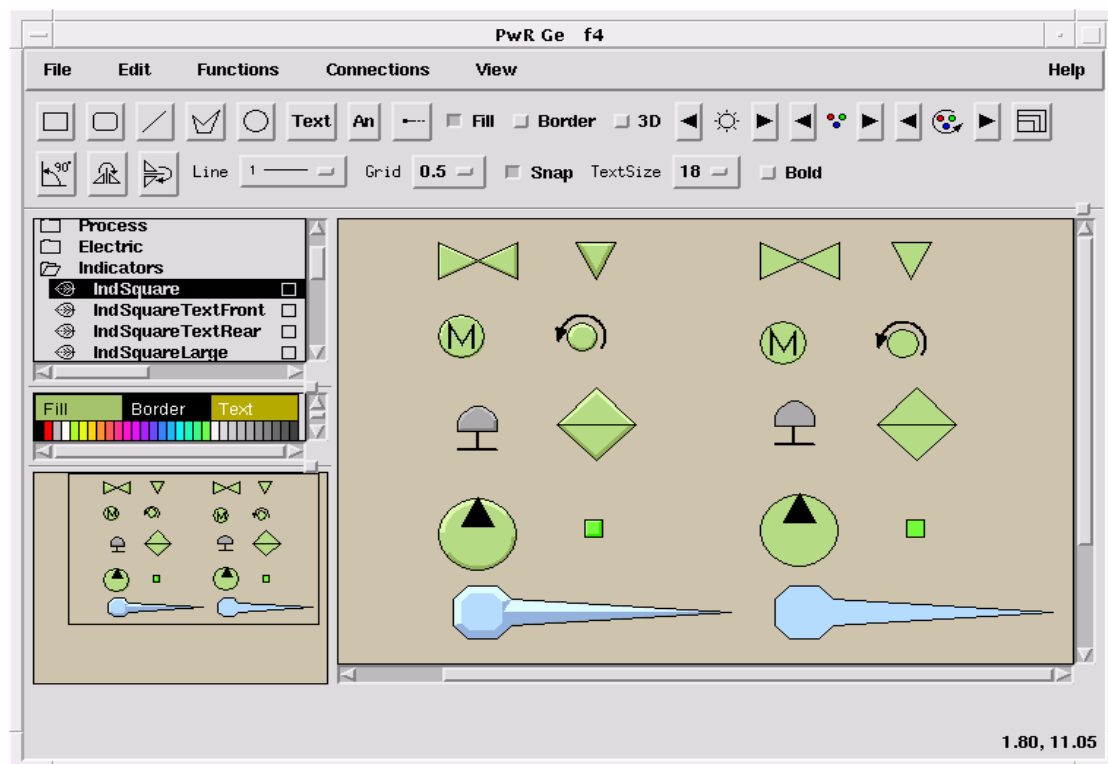
Objekt kan ritas med 3D effekt, dvs med ljusare överkant och mörkare underkant.

Fel!



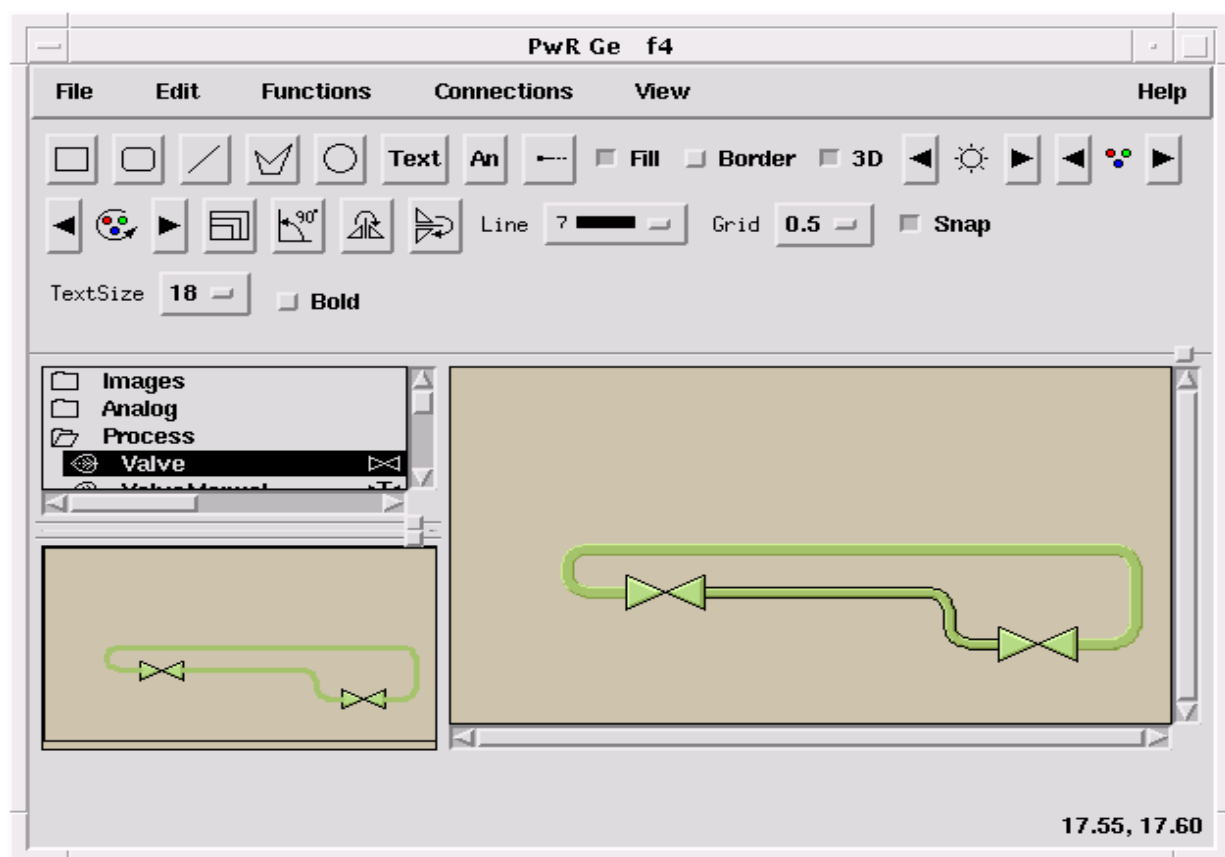
Även komponenter kan ritas i 3D

Fel!



Kopplingar med kantlinjer och 3D

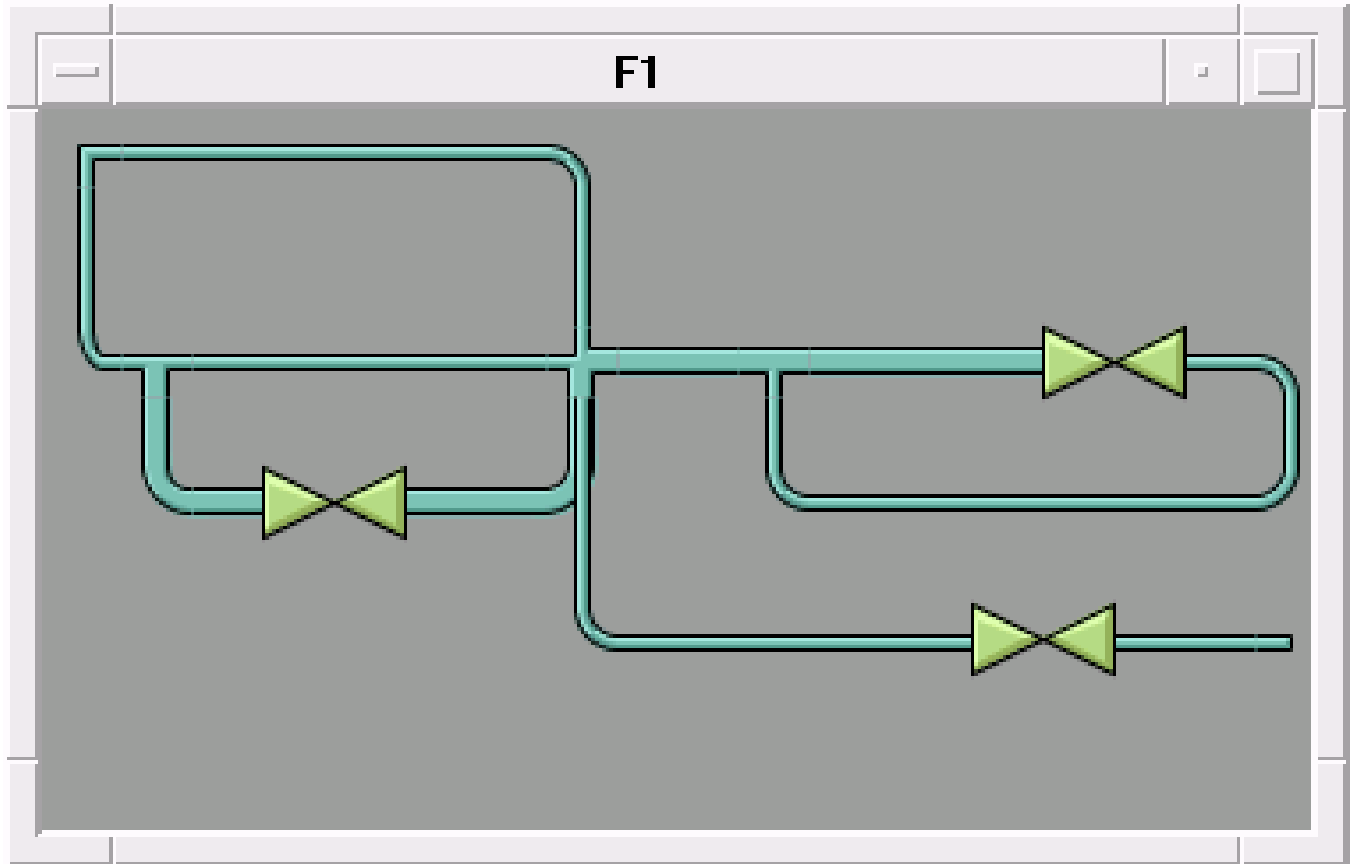
Fel!



”Rörkopplingar”

Ett speciellt objekt ”ConGlue” finns för att binda samman kopplingar av olika tjocklek. ConGlue är ett objekt med 4 kopplingspunkter som anpassar sin form och färg efter de kopplingar som är kopplad till den. Ett ConGlue objekt skapas automatiskt om man drar ut en koppling och släpper den i bakgrunnsarean.

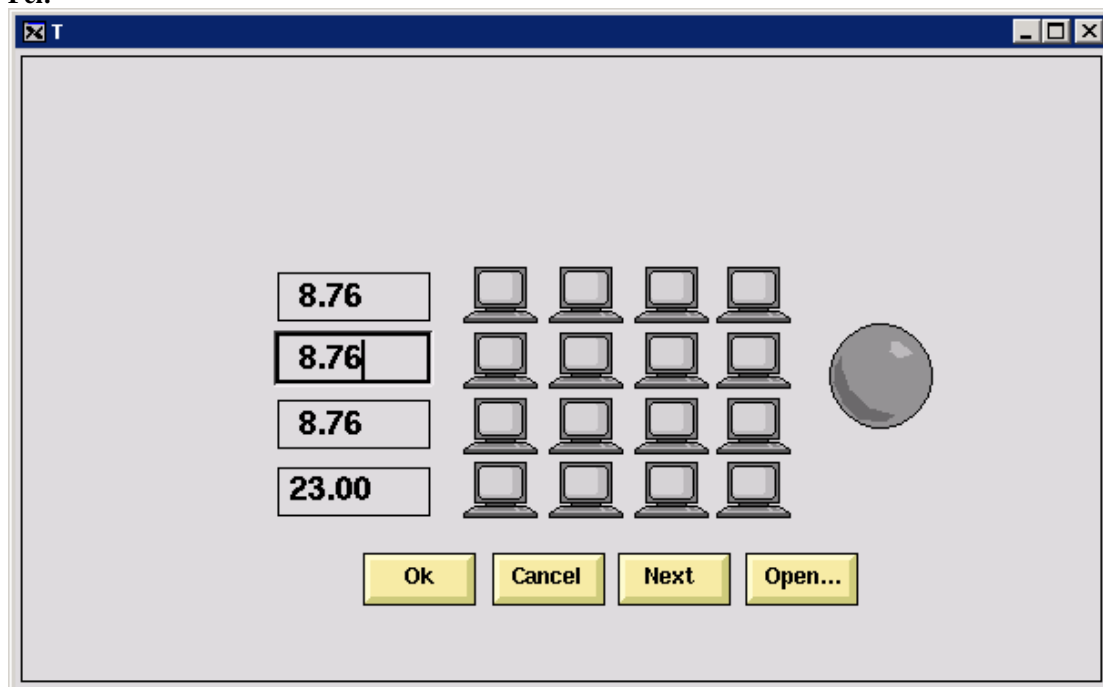
Fel!



Input focus

Inmating sker nu direkt i inmatningsfältet. Det finns även en speciell action för att navigera mellan inmatningsfält med tangentbordet (TAB och pil-tangenterna).

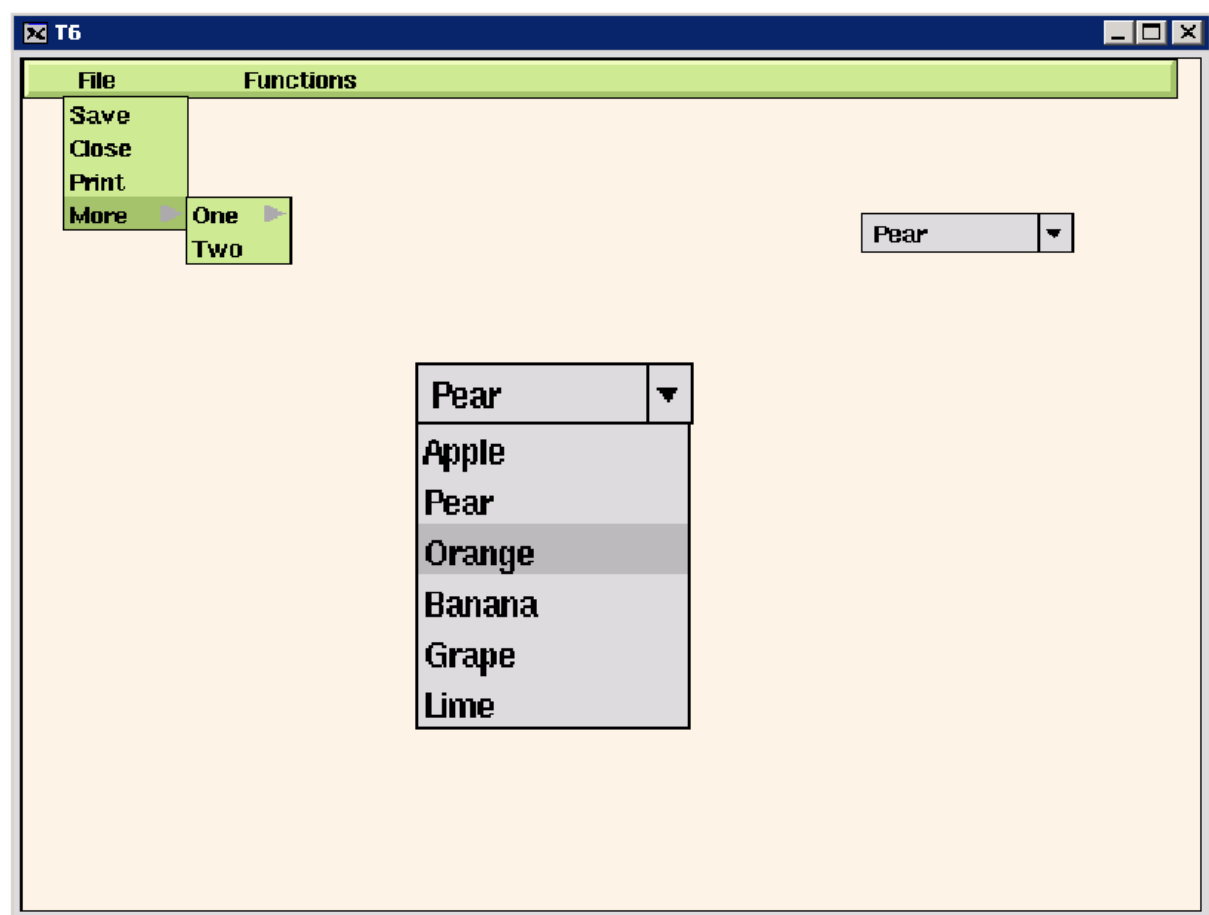
Fel!



Menyer

Nya objekt för att bygga menyer och optionmenyer (comboboxar).

Menyn skapas med subgraferna *Other/Menubar* och *Other/PulldownMenu*, optionmenyn med *Other/OptionMenu*.



Nya klasser

WbEnvironment

Objekt i projektvolymen för att konfigurera ladddatafiler som ska läsas när arbetsbänken startas.

BusConfig

Konfigurerar en QCOM bus i projektvolymen. BusConfig objektet är top objekt på nod sidan, och ska ha NodeConfig objekten för de noder som tillhör bussen som barn.

Distribute

Objekt i projektvolymen för att konfigurera distributören. Detta ersätter många ApplDistribute och GraphDistribute objekt. I detta objekt kan man ange om filer som tillhör ett standard-system, t ex pwrp_alias, \$pwrp_exe/*.pwg, Rt_xtt, ska kopieras eller inte. Objektet läggs under NodeConfig objektet.

SubVolume

SubVolume-objektet, liksom SubVolumeConfig och SubVolumeLoad nedan, har visserligen funnits med sedan V2.7, men inte implementeras förrän nu. Med subvolymen kan man dela upp objektsrymden i en nod, i flera volymer, och därmed i flera databaser. Eftersom endast en åt gången kan arbeta i en databas, är det nödvändigt att dela upp en nod i subvolymen om man vill jobba flera samtidigt. En subvolym konfigureras med SubVolumeConfig och SubVolumeLoad objekt och monteras i rootvolymen med ett MountObject-objekt.

SubVolumeConfig

Objekt i projektvolymen för att konfigurera en subvolym. Objektet läggs på topp-nivån.

SubVolumeLoad

Objekt i projektvolymen för att konfigurera en subvolym. Objektet läggs under NodeConfig-objektet för den nod som ska ladda in volymen.

IiArea

Io-kopierings objekt för Ii (integer input objekt, se nedan).

IoArea

Io-kopierings objekt for Io (integer output objekt, se nedan)

Iv

Integer value. Io-kopierad signal som lagrar ett heltalsvärde i form av en pwr_tInt32.

Ii

Integer input. Signal för inläsning av heltalsvärden från profibus moduler. En Ii signal kopplas till ett ChanIi objekt.

Io

Integer output. Signal för skrivning av heltalsvärden till profibus moduler. En Io signal kopplas till ett ChanIo objekt.

Chanli

Integer input kanal. Konfigurerar inläsning av heltalsvärden på profibus.

Chanlo

Integer output kanal. Konfigurerar skrivning av heltalsvärden på profibus.

Getlv

Hämtar upp värdet på en Iv signal i plckoden. Värdet hämtas som en integer och objektet kopplas till en integer ingång.

Getli

Hämtar upp värdet på en Ii signal i plckoden. Värdet hämtas som en integer och objektet kopplas till en integer ingång.

Getlo

Hämtar upp värdet på en Io signal i plckoden. Värdet hämtas som en integer och objektet kopplas till en integer ingång.

Stolv

Lagrar ett heltalsvärde i en Iv signal. Objektet kopplas till en integer utgång.

Stoli

Lagrar ett heltalsvärde i en Ii signal. Objektet kopplas till en integer utgång.

Stolo

Lagrar ett heltalsvärde i en Io signal. Objektet kopplas till en integer utgång.

CStolv

Villkorlig lagring av ett heltalsvärde i en Iv signal. Objektet kopplas till en integer utgång.

CStoli

Villkorlig lagring ett heltalsvärde i en Ii signal. Objektet kopplas till en integer utgång.

CStolo

Villkorlig lagring ett heltalsvärde i en Io signal. Objektet kopplas till en integer utgång.

Getlp

Hämtar ett heltalsvärde från ett heltals-attribut. Objektet kopplas till en integer ingång.
GetIp i V3.4b som konverterade ett heltals-attribut till analogt värde har bytt namn till GetIpToA.

Stolp

Lagrar ett heltalsvärde i ett heltals-attribut. Objektet kopplas till en integer utgång.
StoIp i V3.4b som konverterade från analogt värde har bytt namn till StoAtoIp.

CStolp

Villkorlig lagring ett heltalsvärde i ett heltals-attribut. Objektet kopplas till en integer utgång.
StoIp i V3.4b som konverterade från analogt värde har bytt namn till CStoAtoIp.

Atol

Konverterar analogt värde till integer med avrundning.

ItoA

Konverterar heltals värde till analog värde.

GetIGeneric

Generiskt get objekt som skapas av plceditorn när man drar ut en koppling från en integer ingång. Konverteras till ett GetIv, GetIi, GetIo eller GetIp objekt beroende på vilken typ av objekt den kopplas till (med Connect funktionen).

StoIGeneric

Generiskt sto objekt som skapas av plceditorn när man drar ut en koppling från en integer utgång. Konverteras till ett StoIv, StoIi, StoIo eller StoIp objekt beroende på vilken typ av objekt den kopplas till (med Connect funktionen).

MaskToD

Konverterar en integer bitmask till digitala signaler.

DtoMask

Konverterar digitala signaler till en bitmask.

EnumToD

Konverterar ett uppräkningsvärde till digitala signaler.

DtoEnum

Konverterar digitala signaler till uppräkningsvärde.

GetDattr, GetAattr, GetSattr, Getlattr

Objekt som används i template plc-koden för en klass för att hämta ett attribut i klassen.

StoDattr, SetDattr, ResDattr, StoDattr, StoAattr, Stolattr, StoSattr, CStoAattr, CStolattr, CStoSattr

Objekt som används i template plc-koden för en klass för att lagra ett värde i ett attribut i klassen.

Föråldrade klasser

GraphDistribute

Har ersatts av Distribute.

SystemDistribute

Eftersom VAXELN inte stöds längre är detta objekt inte aktuellt längre.

DbConfig

Det finns nu endast en volym i varje databas, vilket gör DbConfig-objektet överflödigt. De RootVolumeConfig-objekt som DbConfig-objekten tidigare hade som barn, läggs nu på topp-nivån i projekt-volymen.

Övriga

Andra borttagna klasser är:

AAAnalys, DAnalys, DbDistribute, DirectoryDbConfig, Security, TraceDbConfig, samt ett antal kort och rack objekt för RTP och VME rack (Ai_7436, Ao_7455, etc).

Ändrade klasser

TemplateVärden

Template-objekten för följande klasser har ändrats så att kanal-objekt visas med 2 segment i plc-dokumenten: GetXx, SetXx, ResXx, StoXx, CStoXx.

Template-objekten för DSup och ASup har ändrats så att DetectText visas i plc-dokumenten.

GetIp

GetIp i V3.4b som hämtar ett integer attribute och konverterar till en float, heter nu **GetIpToA**.

Stolp

StoIp i V3.4b som lagrar ett analog värde som integer, heter nu **StoAtoIp**. En annan skillnad är att det analoga värdet nu rundas av. Tidigare höggs det av.

CStolp

CStoIp i V3.4b heter nu **CStoAtoIp**, som i likhet med StoAtoIp rundar av det analoga värdet. Den nya CStoIp lagrar ett heltal.

CArithm

CArithm har utökats med 8 integer ingångar och 8 integer utgångar som refereras med I1-I8 resp IO1-IO8 i koden.

DataArithm

DataArithm har utökats med 6 integer ingångar och 6 integer utgångar som refereras med I1-I6 resp IO1-IO6 i koden.

Ai, Ao, Di, Do, Po, Co, ChanAi, ChanAit, ChanAo, ChanDi, ChanDo, ChanCo

Attributet SigChanCon har tidigare varit en korsreferens, så att signalens SigChanCon har pekat på kanal objektet och kanalens SigChanCon har pekat på signalen. Detta har nu ändras så att enbart signalens SigChanCon pekar på kanalen vid konfigureringen. Kanalens SigChanCon fylls inte i och används inte längre i utvecklingsmiljön, däremot fylls den i i runtime vid initiering av IO't.

Installation

Installation av proview V4.0, liksom även V3.4b och V3.9a, sker med Debians packagehandler dpkg. Proview releasen laddas ner via Proview-hemsidan för aktuell version. Under 'Download' finns länkar till basversionen samt eventuella versioner med patchar.

Gör så här:

- Logga in på process eller operatörs-stationen som ska upgraderas.
- Stoppa proview.
- Gå in som superuser med su.
- Gå till hemmakatalogen för root : cd

- Ändra display om detta behövs.
- Starta mozilla med defaulhemsidan på newton: `$ mozilla newton`
- Gå till proview V4.0 hemsidan genom att välja V4.0 i menyn
- Välj 'Download' i menyn.
- Klicka på den senaste releasen för pwrrt_4.0 och spara på disk.
- Gå ur mozilla.
- Installera pwrrt med: `$ dpkg -i pwrrt_4.0.x-y_i386.deb`, där x-y är nedladdad release.

Procedur för uppgradering

Uppgraderingen görs i 3 steg:

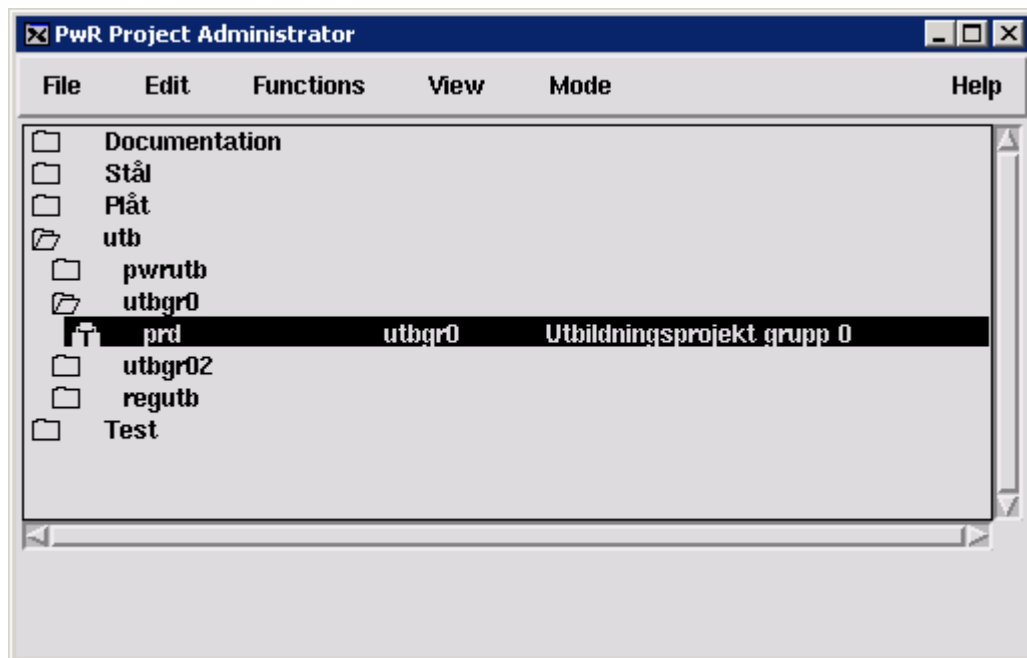
- Ta en kopia av projektet mha wb_adm.
- Exekvera upgrade.sh

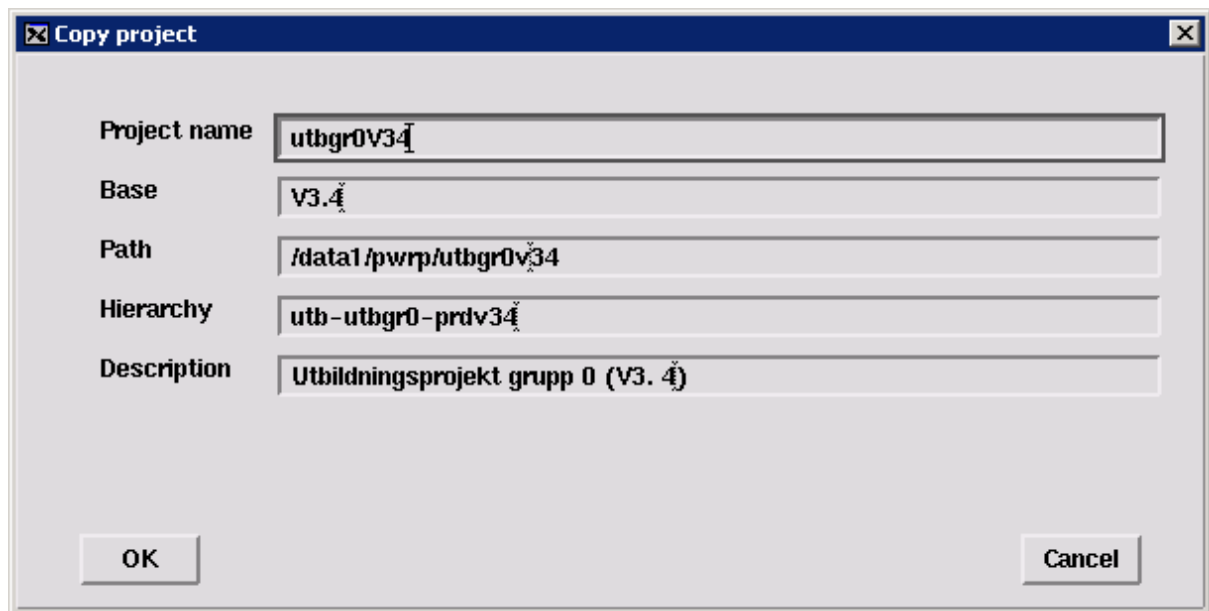
Ta en kopia av projektet

Logga in på newton, gör sdf till projektet och starta projektadministratören.

```
$ pwra
```

Logga in som 'admin' och gå in i editerings mod (*Mode/Edit*). Välj ut projektet och aktivera *Edit/Copy Project*. Nedan skapas en kopia utbgr0v34 som backup för V3.4b projektet, och man uppgraderar det ursprungliga projektet. Man kan naturligtvis lika gärna behålla originalet på V3.4b och uppgradera kopian.



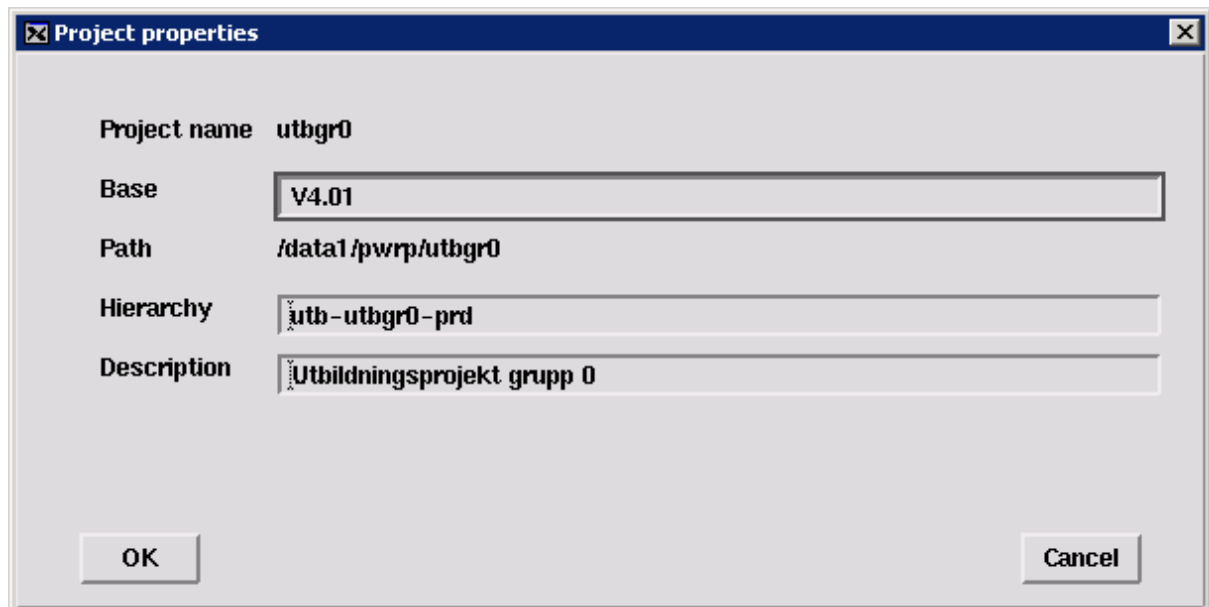


Copy project

Project name	utbgr0V34
Base	V3.4
Path	/data1/pwrp/utbgr0v34
Hierarchy	utb-utbgr0-prdv34
Description	Utbildningsprojekt grupp 0 (V3. 4)

OK Cancel

Ändra version (Base) till V4.01 med *Edit/Project Properties*



Project properties

Project name	utbgr0
Base	V4.01
Path	/data1/pwrp/utbgr0
Hierarchy	utb-utbgr0-prd
Description	Utbildningsprojekt grupp 0

OK Cancel

Gå ur projektadministratören och gör sdf till projektet

```
$ sdf utbgr0
```

upgrade.sh

Exekvera upgrade.sh

```
$ upgrade.sh
```

och kör igenom de olika passen.

dumpdb

Går igenom alla databaser och dumpar varje volym i en egen dumpfil. I V4.0 har ju varje volym sin egen databas och måste då laddas från en separat dumpfil.

Dumpfilen för respektive volym läggs i pwrp_db/'volymnamn'.wb_dmp

userclasses

Skapar en laddatafil från userclasses.wb_load. Laddatafilen får namnet \$pwrp_load/'volymnamn'.dbs.

OBS !

Filen usertypes.wb_load för att definiera typer supportas ej längre av uppgraderings och reload script. Om det finns en usertypes.wb_load måste innehållet i denna flyttas till userclasses.wb_load.

dirvolume

Skapar en directory databas och laddar in dumpfilen för projektvolymer i denna.

cnvdirvolume

Konverterar projektvolymer.

Tar bort föråldrade DbConfig objekt och skapar BusConfig objekt.

cnvdump

Byter namn i dumpfilerna på de klasser som har ändrat namn till V4.0 (GetIp->GetIpToA, StoIp->StoAtoIp, CStoIp->CStoAtoIp)

createvolumes

Skapar databaser för root-volymer och laddar in dump-filerna för respektive volym i dem.

localwb

Skapar en wb_load file for localWb volymen med lokala listdescriptor objekt.

compile

Kompilerar samtliga plc program.

createload

Skapar laddatafiler för rootvolymer.

createboot

Skapar bootfiler för alla noder i projektet.

OBS !

Finns det c-moduler som länkas med plc-programmet ska dessa kompileras först.

convertge

OBS ! Det här passet ska *inte* exekveras vid uppgradering från V3.9.

Konverterar ge grafer från V3.4b till V3.9 format.

Övrigt

Nu återstår att bygga applikationer.

Detta får man göra för hand.

Rensa bort filer från uppgraderingen och filer som inte används i V4.0

\$pwrp_db/*wb_dmp.

\$pwrp_load/ld_vol*.dat

Be systemansvarig att rensa bort mysql-databaserna för V4.0 projektet.