ProviewR

OPEN SOURCE PROCESS CONTROL

Release Notes V5.8

2021 04 16

Table of Contents

Upgrading to ProviewR V5.8.0

This document describes new functions i ProviewR V5.8.0, and how to upgrade a project from V5.7.0 to V5.8.0.

New functions

Graphs from scripts in rt_xtt

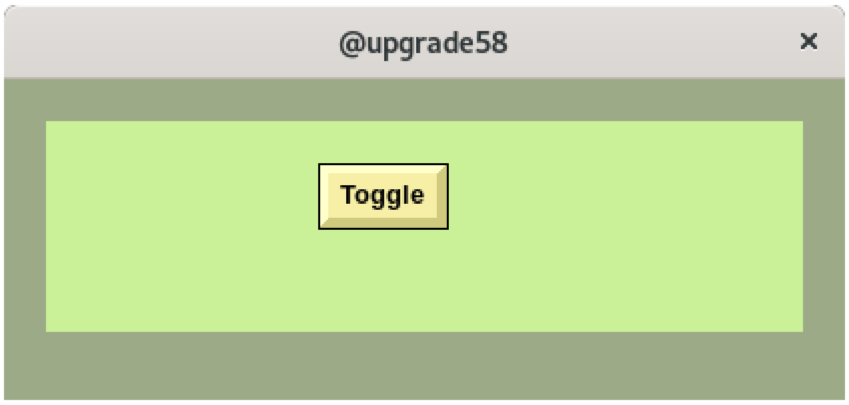
Instead of drawing a graph in the Ge editor, it is now possible to to write a ge script instead. The script can call Ge script functions to draw graphical elements in the graph, but also xtt script functions. This makes it possible to get information from the database and adapt the graph to the current state.

The script graph is opened as an ordinary graph but the graph name is replaced by the script name preceded by a '@', eg 'open graph @myscript'. Scripts are by default read from $pwr_exe or $pwrp_exe, and for any other location the path should be added, eg 'open graph @"$pwrp_login/myscript"'.

Example

A simple script that draws a rectangle and a push button

```
main()
  int id;
  float x1;
  float y1;
  float x2;
  float y2;
  float width;
  float height;

  SetDraw(0);
  SetBackgroundColor(eDrawType_Color66);

  # Draw rectangle
  x1 = 1;
  y1 = 1;
  width = 18;
  height = 5;
  id = CreateRectangle(x1, y1, width, height);
  SetObjectFillColor(id, eDrawType_Color74);
  SetObjectFill(id, 1);
  SetObjectBorder(id, 0);

  # Draw pushbutton
  x1 = 7.5;
  y1 = 2;
  x2 = 10.5;
  y2 = 3.5;
  id = CreateObject("pwr_buttontogglecenter", x1, y1, x2, y2);
  SetObjectAttribute(id, "Text", "Toggle");
  SetObjectAttribute(id, "ToggleDig.Attribute", "H1-
Dv1.ActualValue##Boolean");

  # Set graph size
  SetGraphAttribute("x1", 20.0);
  SetGraphAttribute("y1", 7.0);

  SetDraw(1);
endmain
```



New Ge script functions

BuildGraph()

Build the current graph, i e copy the graph to $pwrp_exe.

ClearAll()

Remote all objects in the graph.

CreateArc()

Create an Arc object.

CreateAxis()

Create an Axis object.

CreateAxisArc()

Create an AxisArc object.

CreateBar()

Create a bar object.

CreateBarArc()

Create a BarArc object.

CreateDsTrend()

Create a DsTrend object.

CreateDsTrendCurve()

Create a DsTrendCurve object.

CreateFastCurve()

Create a DsFastCurve object.

CreateImage()

Create an Image object.

CreateObject()

Create a subgraph object.

CreatePie()

Create a Pie object.

CreatePolyLine()

Create a PolyLine object.

CreateRectangle()

Create a rectangle object.

CreateRectRounded()

Create a rounded rectangle object.

CreateSevHist()

Create a SevHist curve object.

CreateText()

Create a text object.

CreateTrend()

Create a trend object.

CreateXYCurve()

Create an XYCurve object.

DashInsert()

Insert an object into a dashboard cell.

GetGraphName()

Get name of the current graph.

OpenGraph()

Open a graph.

PolyLineAdd()

Add a segment to a polyline.

RotateSelected()

Rotate selected objects.

SaveGraph()

Save the current graph.

SetBackgroundColor()

Set background color.

SetSelectTextBold()

Set bold text on selected objects.

SetSelectTextFont()

Set font on selected objects.

Other script functions

tstlog_open()

Open a test log file.

tstlog_close()

Close a test log file.

tstlog_log()

Print a message to a test log file.

tstlog_vlog()

Print a formated message to a test log file.

getmsg()

Get text for status variable.

tzset()

Set time zone.

sort()

Sort a string array in alphabetical order, or a float or integer array in numerical order.

sin()

Sine function.

cos()

Cosine function.

ODD()

Check if a value is odd.

EVEN()

Check is a value is even.

MAX()

Return the largest of two values.

MIN()

Return the smallest of two values.

Script arrays with dynamic size

A dynamic array is declared without size, eg

```
float darray[];
```

It’s created with zero size and can be incremented by a call to arraypush(). The function arraysize() will return the current size of the array.

arrayclear()

Clear the content of a dynamic array and set size to zero.

arraypush()

Add an element to the back of a dynamic array.

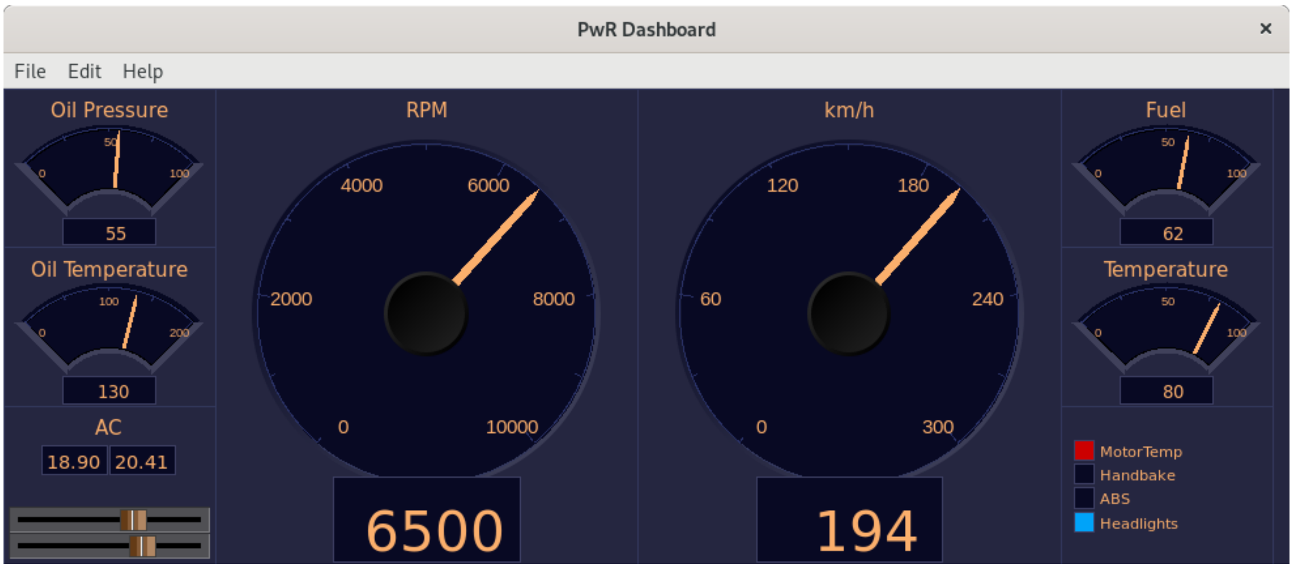
arraysize()

Returns the current size of a dynamic array.

Python3

The Python interface is ported to Python3. It's cannot be executed with Python2 any more.

Dashboard



The dashboard is a simplified graph with a very limited number of building blocks. The building blocks are called dashcells that can display values in the shape of indicators, value fields, bars, trends, gauges etc.

The dashboard can be created in the Ge editor, or in the operator environment.

The dashboard is divided in rows and columns where the cells are positioned. A cell initially has the height of one row and the width of one column, but can be expanded over several rows and columns.

The cell can contain several elements. A bar cell for example, can contain three elements. Each element is connected to an analog attribute that are displayed with a bar. A trend cell can contain two elements and show two curves.

There are three major cell types, analog, digital and object.

Analog dashboard cells

Bar

A bar cell displays an analog value with a bar and a value. I can contain three elements.

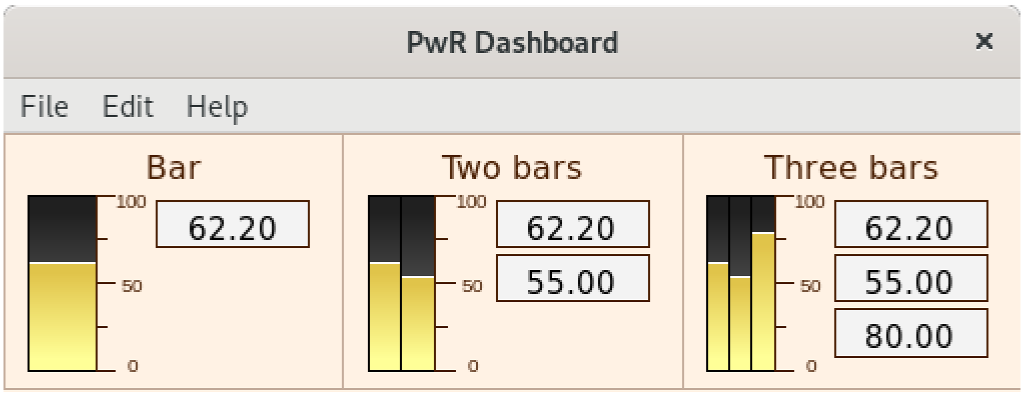
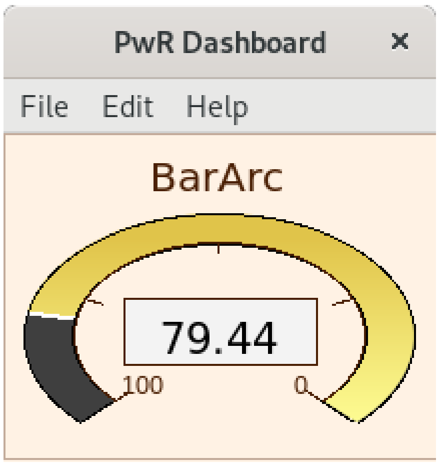
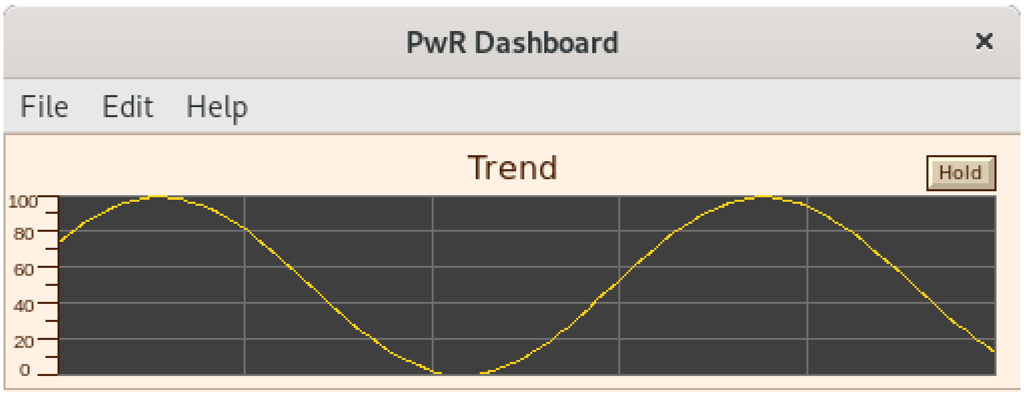


Fig Bar cells with one, two and three elements

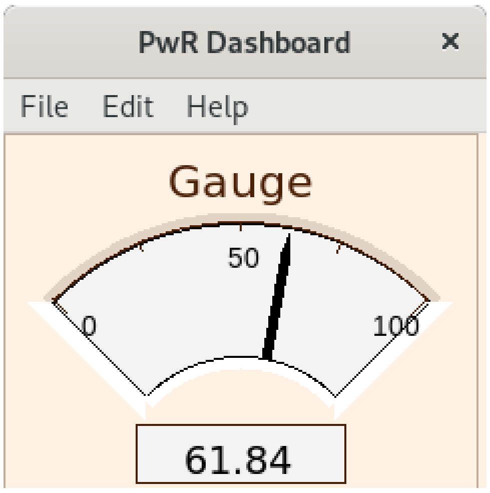
BarArc



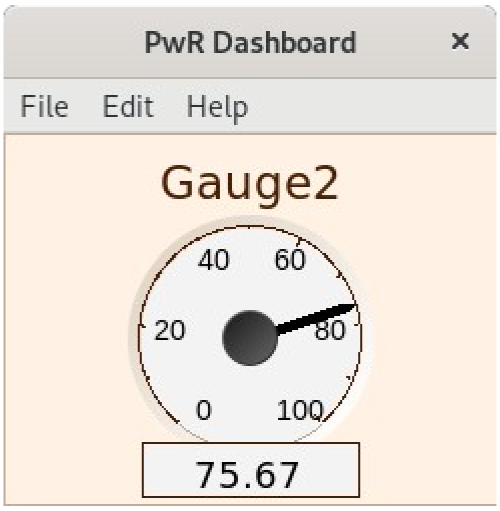
Trend



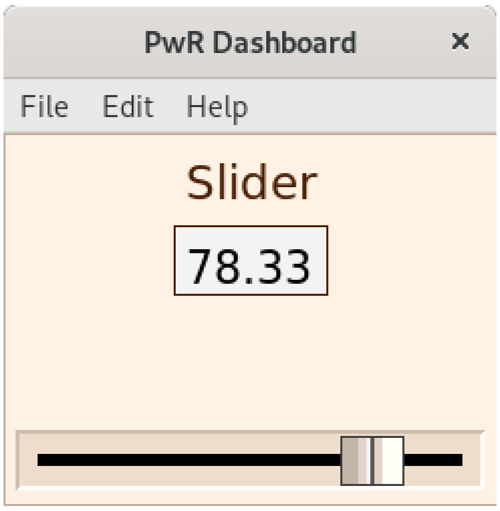
Gauge



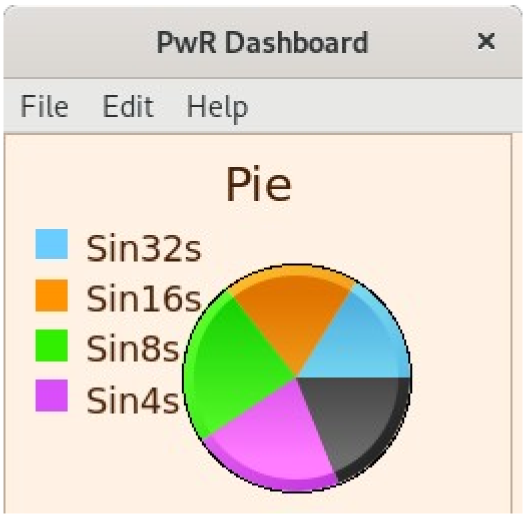
Gauge2



Slider

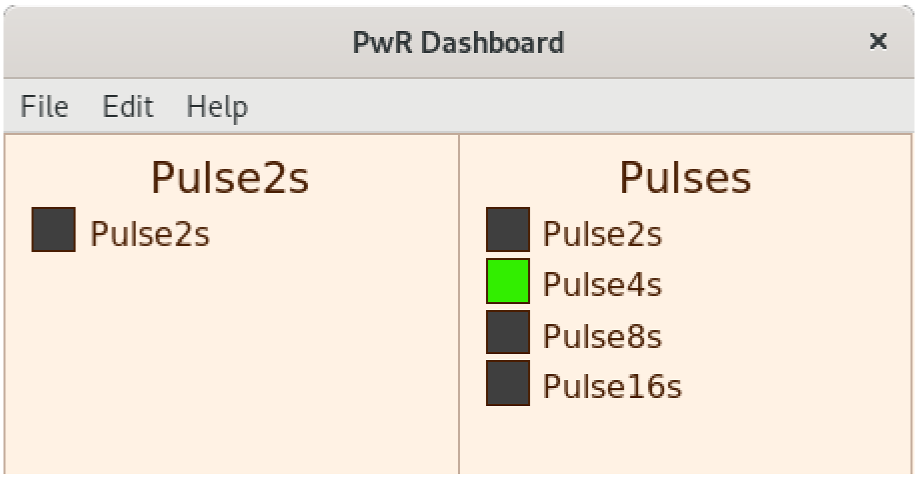


Pie

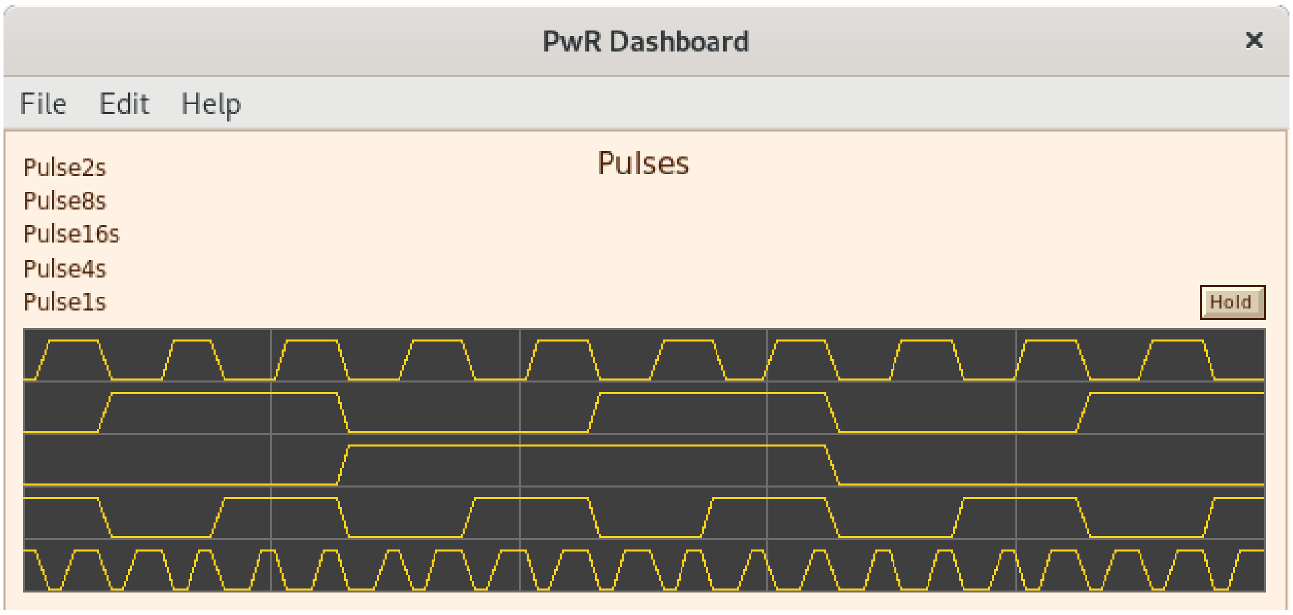


Digital dashboard cells

Indicator



DigitalTrend



Object dash cells

So far object cells are created for DsTrend, DsTrendCurve, SevHist and Sim_SignalGenerator. Two examples are displayed here.

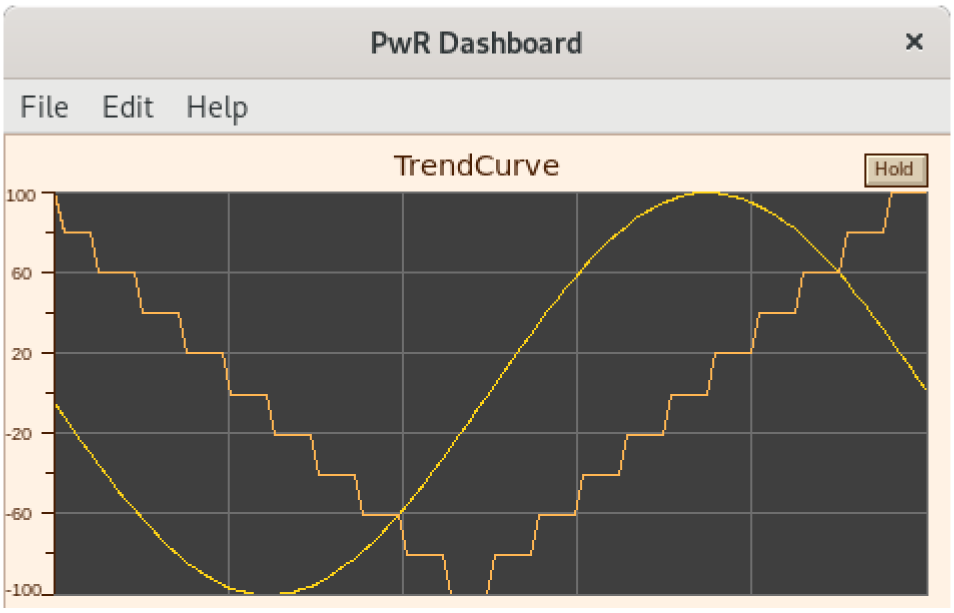
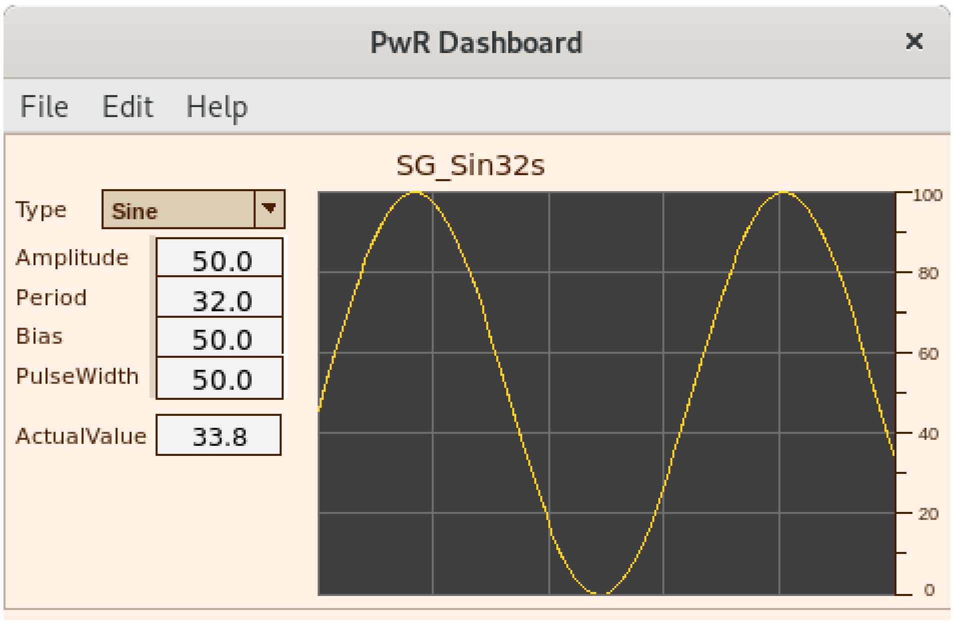


Fig Object cell for DsTrendCurve



Type: Sine
Amplitude: 50.0
Period: 32.0
Bias: 50.0
PulseWidth: 50.0
ActualValue: 33.8

Fig Object cell for Sim_SignalGenerator

Create a dashbord in the Ge editor

A dashboard is created in the editor by setting Dashboard in GraphAttributes to 1. Dashcells can now be created from Other/DashCell in the palette. Select a color theme from File/Colortheme/Select as the cells are draw in color theme colors.

Open the Object Attributes for the cell and select a type in Dash.Type. Connect each element to an attribute in the database and fill in other settings in the Object Attributes.

The size of the dashboard are set in DashRows and DashColumns in GraphAttributes.

When the dashboard is saved, a pwd-file is created on $pwrp_pop, and when it’s built, it’s copied to $pwrp_exe from where it can be opened by rt_xtt.

In rt_xtt the dashboard is opened from Functions/Dashboard/Open or with the command ‘open dashboard ‘name’’.

Create a dashboard in rt_xtt

By selecting an attribute in the navigator, and activating Functions/Dashboard/Insert (Ctrl+D) in the menu, the attribute is added to a dashboard. By default it’s added to the ‘PwR Dashboard’, but if there is a dashboard that is in edit mode, it will be inserted in this dashboard instead. It’s possible to add new attributes to the dashboard until all rows and columns are filled.

For an analog attribute a bar cell is created. This can be changed by entering edit mode in the dashboard from File/Edit (Ctrl+E) in the dashboard menu. Open the cell attributes (double click) and select another type in Dash.Type. Leave the edit mode with Ctrl+E.

When entering edit mode, the Ge editor is opened in restricted mode. Some functionality in edit mode is

- Add. Create an empty cell.
- Delete. Delete cell.
- Copy. Copy a cell.
- Paste. Paste previously copied cells.

- Connect. Connect selected attribute in the navigator to cell.
- Merge. Merge cells. The selected cells will be merged into the one that was first selected.
- From GraphAttributes, the scan time and dashboard size can be modified.

Create a user object dashcell

A dashboard cell for a user object (or any other object) can be created with a Ge script. The script is named dash_'classname'.ge_com and placed on $pwrp_exe. The script should contain the function dash_objectgraph() with the arguments below. All arguments are configurable from the cell attributes editor, except the cell identity.

g	integer	DashCell identity.
elements	integer	Number of elements. 1 for object dashcells.
title	string	Title. By default the object name
time_range	float	Time range than can be used if the cell contains a trend.
direction	int	Direction for bars, trends etc
object	string	Instance object for the cell.
text	string	Configured text.
min_value	float	Min value for axis, trends, bars, sliders etc
max_value	float	Max value for axis, trends, bars, sliders etc

The example below is a script for an Iv object, $pwrp_exe/dash_iv.ge_com.

```
function int dash_objectgraph(int g, int elements, string title,\
 float time_range, int direction, string object, string text, \
 float min_value, float max_value)
  float x1;
  float y1;
  float x2;
  float y2;
  float twidth;
  int id;
  float gx1;
  float gx2;
  float gy1;
  float gy2;
  float gw;
  float gh;
  string attr;

  MeasureObject(g, gx1, gy1, gx2, gy2);
  gw = gx2 - gx1;
  gh = gy2 - gy1;

# Draw header text
  GetTextExtent(title, 3, eFont_LucidaSans, 0, twidth);
  x2 = (gx1 + gx2)/2 - twidth/2;
  y2 = gy1 + 1;
  id = CreateText(title, x2, y2, 3, eFont_LucidaSans, 0, \
    eDrawType_CustomColor5);
  DashInsertObject(g, id);

# Draw value field
  x1 = gx1 + 1.0;
  y1 = gy1 + 1.5;
  x2 = gx2 - 1.0;
  y2 = y1 + (x2 - x1) / 2.5;
  id = CreateObject("pwrct_valuelargecenter", x1, y1, x2, y2);
  attr = object + ".ActualValue##Int32";
  SetObjectAttribute(id, "Value.Attribute", attr);
  SetObjectAttribute(id, "Value.Format", "%d");
  DashInsertObject(g, id);

endfunction
```

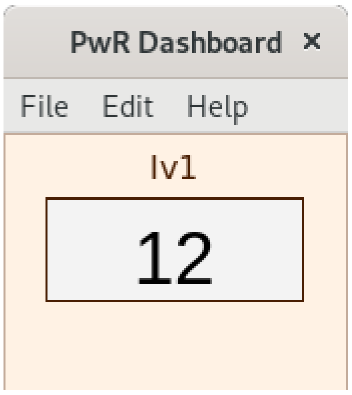


Fig The user object dashcell

Start runtime as another node

The -n option to rt_ini makes it possible to start a runtime environment that is configured for another node on the current node. The Qcom bus id though has to be the same.

```
> rt_ini -n pwrtest01a
```

will start the node pwrtest01a on the current node.

It is also possible to redirect qcom connections to other nodes with the alias statement in pwrp_alias.dat.

```
alias pwrtest01b steel-arrow 10.255.0.98
```

will redirect the connection to pwrtest01b to the node steel-arrow with ip address 10.255.0.98.

Display a DsTrend object in a graph

New Ge component to view a DsTrend curve in a Ge graph. The component is found under Analog/ DsTrend in the palette.

The object can is connected to one or two DsTrend objects in the database and can display two curves.

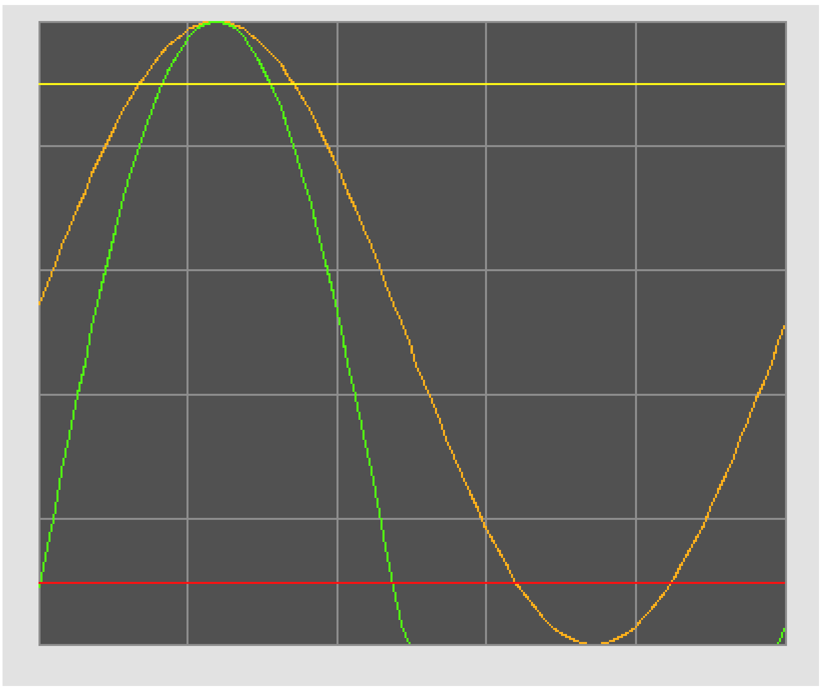


Fig DsTrend curve displayed in a Ge graph

Display a DsTrendCurve object in a graph

New Ge component to view a DsTrendCurve curve in a Ge graph. The component is found under Analog/DsTrendCurve in the palette.

The object can is connected to a DsTrendCurve object in the database and can display two curves.

Display a history curve in a graph

New Ge component to view a SevHist curve in a Ge graph. The component is found under Analog/SevHist in the palette.

The object can is connected to one or two SevHist object in the database and can display two curves.

In case the curve should be viewed from a sev server, where the SevHist object might not be present, the attributes name and history server node can be supplied instead.

The curve can be displayed as a snapshot or cyclically updated.

New Types and Classes

PulseTrainM

Plc function object with pulse train outputs with different cycle times: 128 s, 64 s, 32 s, 16 s, 8 s, 4s 2 s, 1 s, 500 ms, 250 ms, 125 ms, 62.5 ms, 31.2 ms and 15.6 ms. In comparison with PulseTrain the pulse times in PulseTrainM are a multiple of each other and synchronized.

DtoI

Plc function object to convert a digital signal to integer.

LightATv

Absolute time value that is not IO copied and only contains the basic methods. When used in a plc program considerations about thread safety and execution order has to be taken.

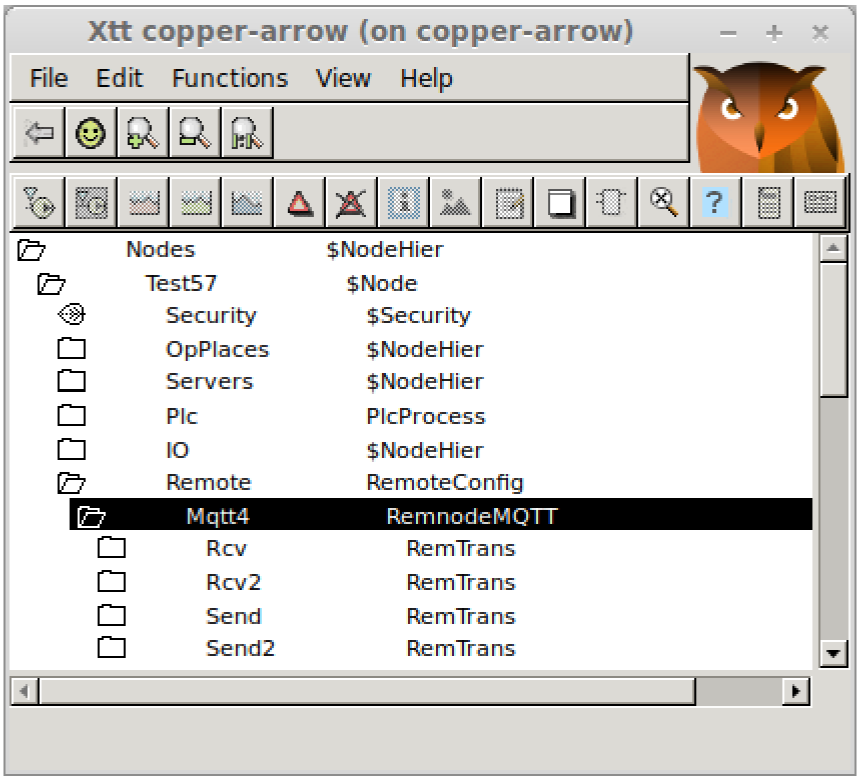
LightDTv

Delta time value that is not IO copied and only contains the basic methods. When used in a plc program considerations about thread safety and execution order has to be taken.

Additions in V5.8.1

Remote transactions with MQTT

Remote transactions can now be sent and received via MQTT with the RemnodeMQTT object.



```
Nodes            $NodeHier
  Test57           $Node
    Security         $Security
    OpPlaces         $NodeHier
    Servers          $NodeHier
    Plc              PlcProcess
    IO               $NodeHier
    Remote           RemoteConfig
      Mqtt4            RemnodeMQTT
        Rcv              RemTrans
        Rcv2             RemTrans
        Send             RemTrans
        Send2            RemTrans
```

Fig Remote configuration with MQTT

The communication is configured with a RemnodeMQTT object. The MQTT server, topics for publishing and subscribing are specified here.

The specification of topics differs if the remote header is disabled or not.

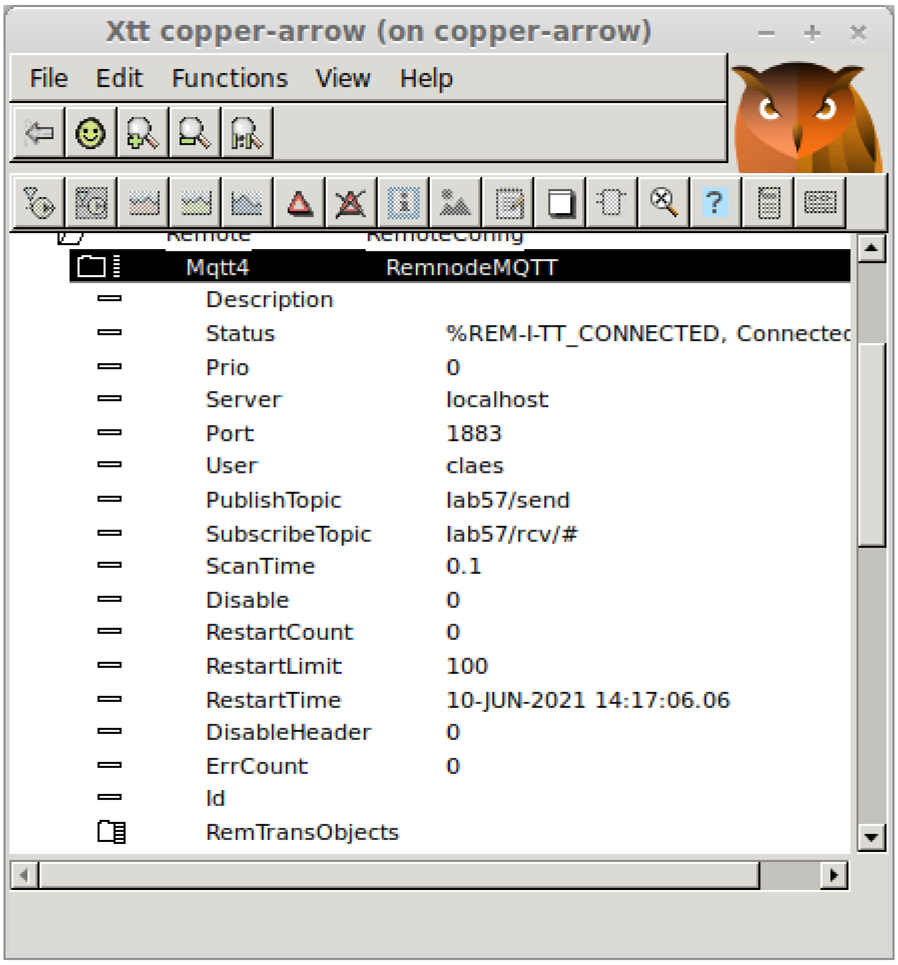
If the header is present

- Sending: publishing is made with the topic in PubishTopic in the RemnodeMQTT. Address[0] and Address[1] in the RemTrans object is used to match RemTrans objects.
- Receiving: Subscriptions are med with the topic in SubscribeTopic in the RemnodeMQTT object. The message is directed to the RemTrans with matching Address[0] and Address[1].

If header is disabled

- Sending: publishing is made with the topic in RemTrans.TransName
- Receiving: A generic topic is set in SubscribeTopic in the RemnodeMQTT object, eg 'lab57/

rcv/#'. A more narrow topic is set in RemTrans.TransName, eg 'lab57/rcv/msg1'.



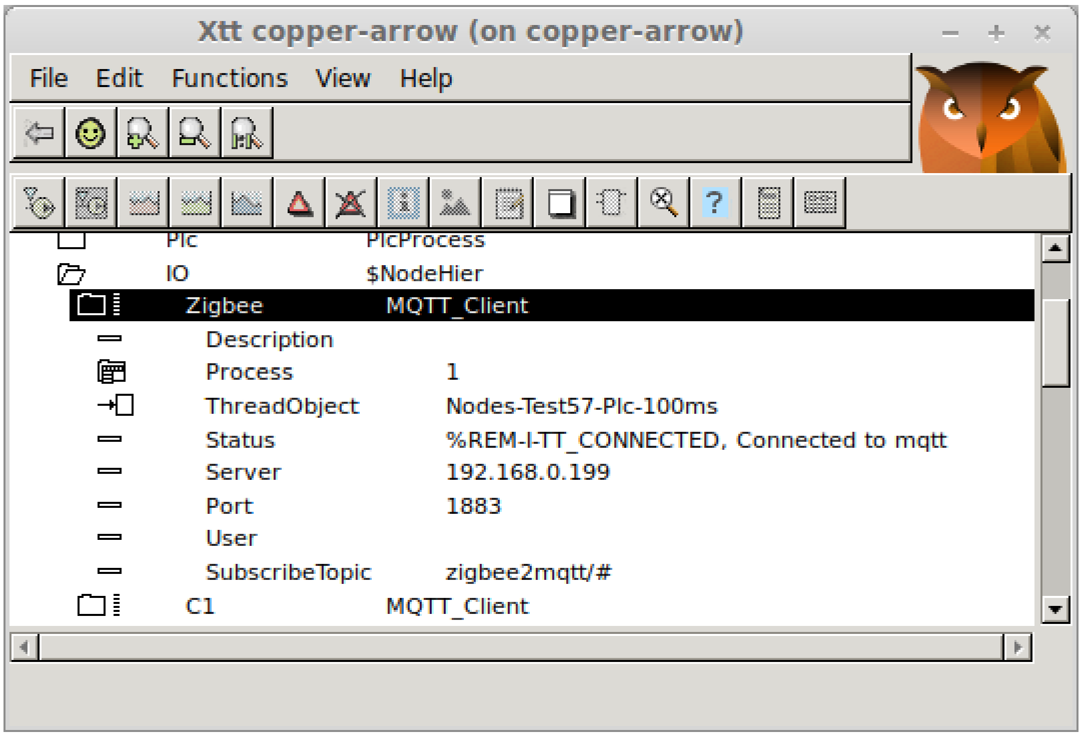
```
Mqtt4              RemnodeMQTT
    Description
    Status           %REM-I-TT_CONNECTED, Connected
    Prio             0
    Server           localhost
    Port             1883
    User             claes
    PublishTopic     lab57/send
    SubscribeTopic   lab57/rcv/#
    ScanTime         0.1
    Disable          0
    RestartCount     0
    RestartLimit     100
    RestartTime      10-JUN-2021 14:17:06.06
    DisableHeader    0
    ErrCount         0
    Id
    RemTransObjects
```

Fig RemnodeMQTT object

MQTT IO

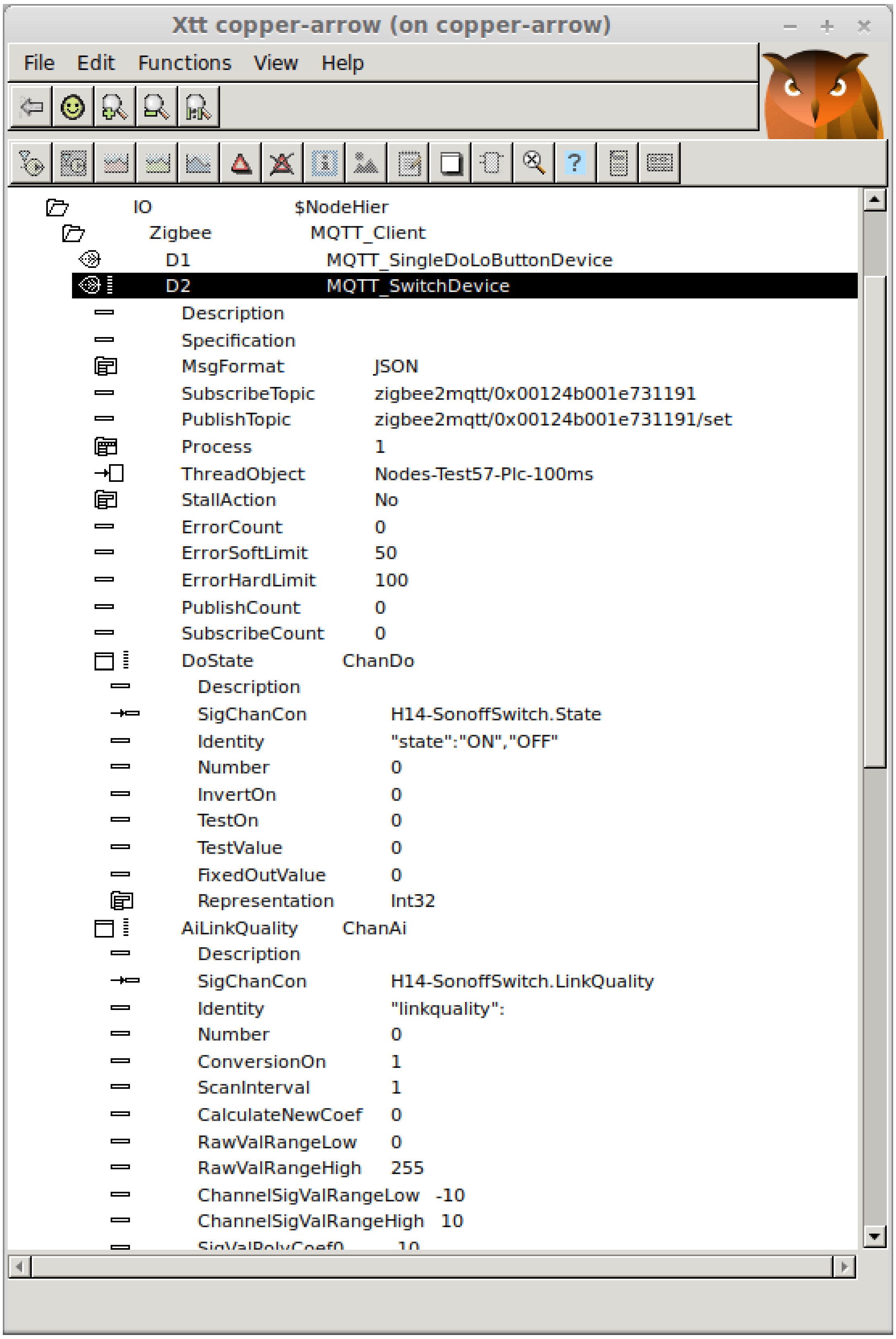
The MQTT IO can read values published on an MQTT server into insignals and publish values of outsignals to an MQTT server.

The configuration is done with a MQTT_Client object and MQTT_Device objects.


Xtt copper-arrow (on copper-arrow)
File Edit Functions View Help

```
Plc             PlcProcess
IO              $NodeHier
  Zigbee          MQTT_Client
    Description
    Process           1
    ThreadObject      Nodes-Test57-Plc-100ms
    Status            %REM-I-TT_CONNECTED, Connected to mqtt
    Server            192.168.0.199
    Port              1883
    User
    SubscribeTopic    zigbee2mqtt/#
  C1              MQTT_Client
```

Fig MQTT_Client


Xtt copper-arrow (on copper-arrow)
File Edit Functions View Help

```
IO                 $NodeHier
  Zigbee             MQTT_Client
    D1                 MQTT_SingleDoLoButtonDevice
    D2                 MQTT_SwitchDevice
      Description
      Specification
      MsgFormat            JSON
      SubscribeTopic       zigbee2mqtt/0x00124b001e731191
      PublishTopic         zigbee2mqtt/0x00124b001e731191/set
      Process              1
      ThreadObject         Nodes-Test57-Plc-100ms
      StallAction          No
      ErrorCount           0
      ErrorSoftLimit       50
      ErrorHardLimit       100
      PublishCount         0
      SubscribeCount       0
      DoState          ChanDo
        Description
        SigChanCon           H14-SonoffSwitch.State
        Identity             "state":"ON","OFF"
        Number               0
        InvertOn             0
        TestOn               0
        TestValue            0
        FixedOutValue        0
        Representation       Int32
      AiLinkQuality    ChanAi
        Description
        SigChanCon           H14-SonoffSwitch.LinkQuality
        Identity             "linkquality":
        Number               0
        ConversionOn         1
        ScanInterval         1
        CalculateNewCoef     0
        RawValRangeLow       0
        RawValRangeHigh      255
        ChannelSigValRangeLow  -10
        ChannelSigValRangeHigh  10
```

Fig Switch object based on MQTT_Device with a Do and an Ai channel

Home automation with zigbee2mqtt

For those interested in home automation I just want to mention zigbee2mqtt that makes it possible to access a large range of zigbee devices. Zigbee2mqtt is flashed to a usbstick that acts a zigbee coordinator. A server program is run on RasberryPi that presents the zigbee communication on an

MQTT server in json format. This can then be accessed by the ProviewR MQTT IO.

ProviewR Mqtt server

The ProviewR Mqtt server is a program that can answer request via MQTT about attribute values and history data. It is also possible to setup subscriptions and set values.

The server is configured with a MqttServer object in the node hierarchy. When this object is created the server program rt_mqtt_server will start.

Requests can be put to a specific topic on an MQTT broker. The topic is configured in the MqttServer object and is 'proviewr/server' by default. The request should be on json format and contain “action” that specifies the type of request. The action can be get, set, subscribe, closesub, sublist, closesublist or history.

Get

Get the value of an attribute.

Request

action	Should be “get”.
attribute	Full name of the attribute.
reply	MQTT topic where the reply should be published.

Reply

status	Return status.
value	Attribute value.

Example

Request

 {"action":"get","attribute":"H16-Av1.ActualValue","reply":"repl/get"}

Reply

{“status”:141459465,"value":45.9577}

Set

Set the value of an attribute.

Request

action	Should be “set”.
attribute	Full name of the attribute.
reply	MQTT topic where the reply should be published.

Reply

status	Return status.

Example

Request

{"action":"set","attribute":"H16-Av1.ActualValue","reply":"repl/get"}

Reply

{“status”:141459465}

Subscribe

Set up a subscription of an attribute. The value of the attribute should cyclically be sent to the reply topic until the duration time has elapsed, or the subscription is closed with a closesub message.

Request

action	Should be “subscribe”.
attribute	Full name of the attribute.
cycle	Cycle time in seconds.
duration	Max duration of the subscription.
reply	MQTT topic where the reply should be published.

Reply

subref	Subscription reference.
status	Not yet implemented.
value	Attribute value.

Example

Request

{"action":"subscribe","attribute":"H16-Av1.ActualValue","cycle":"1.0","duration":"300.0","reply":"repl/subscribe"}

Reply

{'subref': 1, 'status': 0, 'value': -14.98953}

Closesub

Close a subscription started with the subscribe action.

Request

action	Should be “closesub”.
subref	Subscription reference.

No reply is sent.

Example

Request

{"action":"closesub","subref”:5}

Sublist

Set up a subscription of a number of attributes. The values of the attributes should cyclically be sent to the reply topic until the duration time has elapsed, or the subscription is closed with a closesublist message.

Request

action	Should be “sublist”.
attribute	Array of index and full name of the attribute.
cycle	Cycle time in seconds.
duration	Max duration of the subscription.
reply	MQTT topic where the reply should be published.

Reply

subref	Subscription reference.
a	Array of index and attribute value.

Example

Request

{"action":"sublist","cycle":"1.0","duration":"15.0","reply":"repl/sublist","attribute":[{1,"H16-Av1.ActualValue"},{2,"H16-Av2.ActualValue"},{3,"H16-Av4.ActualValue"}]}

Reply

{"subref":2,"a":[{"idx":1,"value":39.3397},{"idx":2,"value":39.3397},{"idx":3,"value":42.1814}]}

Closesub

Close a subscription started with the subscribe action.

Request

action	Should be “closesublist”.
subref	Subscription reference.

No reply is sent.

Example

Request

{"action":"closesublist","subref”:2}

History

Get process history for an attribute.

Returns process history from a sev database.

Request

action	Should be “history”.
server	Server from which the history should be fetched.
object	Full object name.
attribute	Attribute name.
from	Start time for history data, eg '20-JUN-2021 12:00:00'.
to	End time for history data, eg '21-JUN-2021 12:00:00'. To get the most recent data 'now' can be used for the current time, and a delta time can be given in 'from', eg 'to':'now' and 'from':'1 00:00:00'.
maxpoints	Max number of points that should be returned.
reply	MQTT topic where the reply should be published.

Reply

status	Status of the action.
values	Array of attribute values.
time	Array of time for the values.

Example

Request

{"action":"history","server":"localhost","object":"H1-Av1",”attribute”:”ActualValue”"reply":"repl/history","from":”0:15:0”, “to”:”now”,”maxrows”:6}

Reply

{“status":1315888905,"values":[-96.1745, -34.2858, 82.5293, 99.5304, 96.0948, 56.3516],”time”:["14-JUN-2021 15:08:48.00","14-JUN-2021 15:09:09.00","14-JUN-2021 15:09:17.00","14-JUN-2021 15:09:23.00","14-JUN-2021 15:09:38.00","14-JUN-2021 15:09:59.00"]}

Eventhist

Get alarm and event history for a table defined by a SevHistEvent object.

Returns event history from a sev database.

Request

action	Should be “eventhist”.
server	Server from which the history should be fetched.
object	Name or identity of SevHistEvent object.
eventtype	Mask for event types that should be searched for. Optional. 1: Ack, 2: Block

	4: Cancel 8: CancelBlock 16: Missing 32: Reblock 64: Return 128: Unblock 256: InfoSuccess 512: Alarm 1024: MaintenanceAlarm 2048: SystemAlarm 4096: UserAlarm1 8192: UserAlarm2 16384: UserAlarm3 32768: UserAlarm4 65536: Info
eventprio	Mask for event priorities that should be searched for. Optional. 1: Prio A 2: Prio B 4: Prio C 8: Prio D
text	Event text with wild card that should be search for. Optional.
name	Event name with wild card that should be searched for. Optional.
from	Start time for history data, eg '20-JUN-2021 12:00:00'.
to	End time for history data, eg '21-JUN-2021 12:00:00'. To get the most recent data 'now' can be used for the current time, and a delta time can be given in 'from', eg 'to':'now' and 'from':'1 00:00:00'.
options	Mask that states which values should be returned. Optional. 1: Time 2: Event type 4: Event priority 8: Event text 16: Event name 32: Event identity
maxpoints	Max number of points that should be returned.
reply	MQTT topic where the reply should be published.

Reply

status	Status of the action.
time	Array of event time.
type	Array of event type.
prio	Array of event priority.
text	Array of event text.
name	Array of event name.
id_nix	Array of event identity, nix.
id_idx	Array of event identity, idx.

Example

Request

{"action":"eventhist","server":"localhost","object":"H1-SevHistEvents","reply":"repl/eventhist","from":”7 0:0:0”, “to”:”now”,”maxrows”:10}

Reply

{"status":135888905,"time":["07-JUN-2021 15:06:21.00","07-JUN-2021 15:06:55.00","07-JUN-2021 15:11:56.00","07-JUN-2021 15:11:56.00","07-JUN-2021 15:11:56.00","07-JUN-2021 15:11:56.00","07-JUN-2021 15:11:58.00","07-JUN-2021 15:11:58.00","07-JUN-2021 15:12:02.00","07-JUN-2021 15:12:02.00"],"type":[256,7,64,64,64,64,64,64,7,7],"text":["System status error, node copper-arrow","","Dv 8 alarm","Dv 7 alarm","Dv 2 alarm","Dv 1 alarm","Dv 10 alarm","Dv 9 alarm","Dv 10 return","Dv 9 return"]}

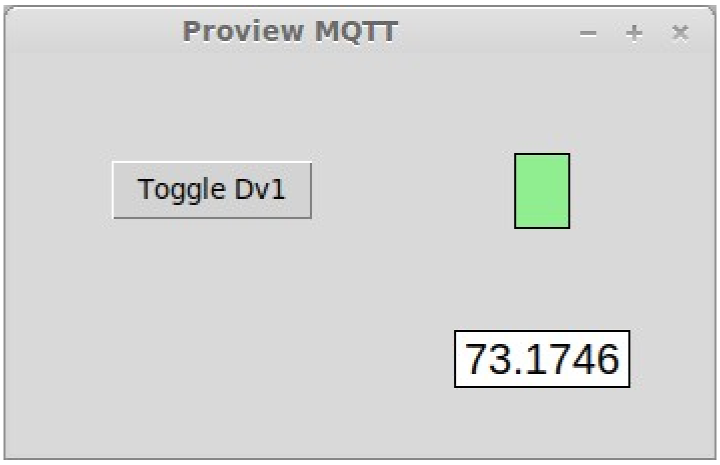
Python example 1

This example fetches the value for an attribute with the 'get' action.

```
#!/usr/bin/python3
#
import paho.mqtt.client as mqtt
import sys
import time
from datetime import datetime
import json

# Print reply
def on_message(client, userdata, message):
    reply = json.loads(str(message.payload.decode("utf-8")))
    value = reply['value']
    print("Reply:", reply)
    print("Value:", value)

# Connect to MQTT on localhost
client = mqtt.Client('MyClient')
client.username_pw_set('pwrp','pwrp')
client.on_message = on_message
client.connect('localhost')

# Send request and subscribe on reply
client.subscribe("repl/get", 1);
client.publish('proviewr/server', '{"action":"get","attribute":"H16-
Av1.ActualValue","reply":"repl/get"}')

# Wait for reply
for i in range (0, 3):
  client.loop_start()
  time.sleep(1)
  client.loop_stop()
```

Python example 2

Simple graph with a pushbutton, an indicator and a value field.

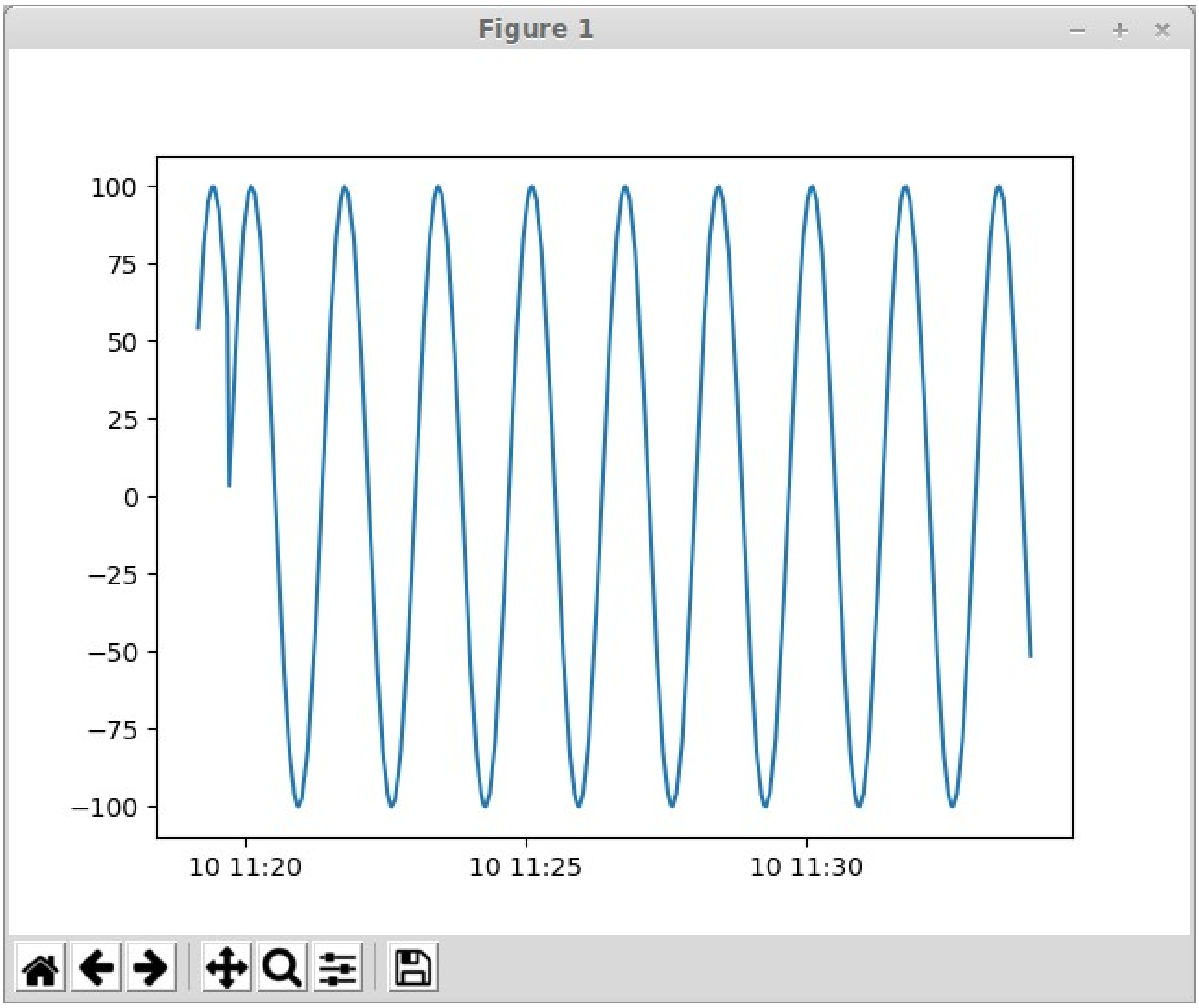


```
#!/usr/bin/python3
#
from tkinter import *
import paho.mqtt.client as mqtt
import sys
import time
from datetime import datetime
import json
import random

def on_closing():
    global subref

    # Close subsciptions
    if subref != 0:
        client.publish('proviewr/server',
'{"action":"closesublist","subref":' + str(subref) + '}')

    window.destroy()


def on_message(client, userdata, message):
    global val1
    global val2
    global subref
    global set_reply
    global sublist_reply

    if message.topic == sublist_reply:
        data = json.loads(str(message.payload.decode("utf-8")))
        subref = data['subref']
        val1 = data['a'][0]['value']
        val2 = data['a'][1]['value']

    if message.topic == set_reply:
        pass

# Button click callback
def button_click_cb():
    global set_reply

    client.subscribe(set_reply, 1);
    if val2 == 0:
        client.publish('proviewr/server', '{"action":"set","attribute":"H17-
Dv1.ActualValue","value":"1","reply":"' + set_reply + '"}')
    else:
        client.publish('proviewr/server', '{"action":"set","attribute":"H17-
Dv1.ActualValue","value":"0","reply":"' + set_reply + '"}')

# Cyclic scan function
def scan():
    global sub1_old
    global sub2_old
    global val1
    global val2

    if val2 != sub2_old:
        if val2 == 1:
            dv1_label["bg"] = "lightgreen"
        else:
            dv1_label["bg"] = "black"
        sub2_old = val2

    if val1 != sub1_old:
        av1_label["text"] = val1
        sub1_old = val1

    window.after(500, scan)

# Create window
window = Tk()
window.title("Proview MQTT")
window.geometry('350x200')

# Create unique name and topics to be able to run serveral instances
rand = str(random.randint(1,999999))
name = 'MqttTest' + rand
sublist_reply = 'repl/' + rand + '/sublist'
set_reply = 'repl/' + rand + '/set'

# Create button
button = Button(window, text="Toggle Dv1", command=button_click_cb,
bg="lightgray")
button.grid(column=0, row=0, padx=50, pady=50)

# Create indicator label
dv1_label = Label(window, width=3, height=2, bg="black", borderwidth=1,
relief="solid")
dv1_label.grid(column=1, row=0, padx=50, pady=50)

# Create value label
av1_label = Label(window, width=7, bg="white", borderwidth=1,
relief="solid",
font=("Helvetica",16))
av1_label.grid(column=1, row=2, padx=20, pady=0)

# Attach MQTT
client = mqtt.Client(name)
client.username_pw_set('pwrp','pwrp')
client.on_message = on_message
client.connect('localhost')
```

```
# Set up subscriptions
client.subscribe(sublist_reply, 1);
client.publish('proviewr/server',
'{"action":"sublist","cycle":"1.0","duration":"150.0","reply":"' +
sublist_reply + '","attribute":[{1,"H17-Av1.ActualValue"},{2,"H17-
Dv1.ActualValue"}]}')

subref = 0
sub1_old = -1
sub2_old = -1
val1 = 0
val2 = 0

window.protocol("WM_DELETE_WINDOW", on_closing);

client.loop_start()
scan()
window.mainloop()
```

Python example 3

Drawing a history curve with matplotlib



```
#!/usr/bin/python3
#
import paho.mqtt.client as mqtt
import sys
import time
from datetime import datetime
import json
import matplotlib.pyplot as plt
from datetime import datetime

def on_log(client, userdata, level, buf):
    print("log: ",buf)

def on_message(client, userdata, message):
    print("message received ", datetime.now(),
str(message.payload.decode("utf-8")), flush=True)
    data = json.loads(str(message.payload.decode("utf-8")))

    # Convert time strings to datetime objects
    t = []
    for dt in data['time']:
        t.append(datetime.strptime(dt+'0000', '%d-%b-%Y %H:%M:%S.%f'))

    # Plot the curve, use drawstyle='steps-pre' for digital signals
    plt.plot(t, data['values'], label='Diff')
    plt.show()


# Connect to MQTT server
client = mqtt.Client('Claes')
client.username_pw_set('pwrp','pwrp')
client.on_message = on_message
client.connect('localhost')

# Subscribe to reply
client.subscribe("repl/history", 1)

# Send history request
client.publish('proviewr/server',
'{"action":"history","reply":"repl/history","server":"localhost","object":"H
1-
Av1","attribute":"ActualValue","from":"0:15:0","to":"now","maxrows":2000}')

for i in range (0, 3):
  print("Loop");
  client.loop_start()
  time.sleep(1)
  client.loop_stop()
```

Upgrade procedure

The upgrading has to be done from any V5.7. 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->V5.7->V5.8

Enter the administrator and change the version of the project to V5.8.0. Save and close the administrator.

Enter the directory volume and save.

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.