PG2 Network Helper

Contents

Initiation

This controls can be used to do simple networking tasks in 800xA PG2 graphic displays. Due the nature of the PG2 graphic system the usage of this controls is limited to single action network operations.

Disclaimer

The author of this program is not liable for damage to software or hardware or property damage incurred by the use of the program.

License

Prerequisites / Compatibility

800xA 6.0 or 6.1

Installation

On every node where you want to use the network helper controls:

- Run **install.bat** as Administrator
- Close every workplace and graphics builder on this node
- Make sure no graphics builder instance is still running in the background, as it often do
- Re-open the workplaces

(The first workplace started on this node reads all graphic primitives to the cache.)
On nodes without the network helper controls installed the graphic display will display a crossed purple rectangle instead of the controls.

Uninstallation

On every node where the network helper controls are installed:

- Close every workplace and graphics builder on this node.
- Make sure no graphics builder instance is still running in the background, as it often do
  (The dll file is write protected if one workplace or graphics builder is still running)
- Run **uninstall.bat** as Administrator

Update

- Close every workplace and graphics builder on this node.
- Make sure no graphics builder instance is still running in the background, as it often do
  (The dll file is write protected if one workplace or graphics builder is still running)
- Replace the old **c:\Program Files (x86)\ABB 800xA\Base\bin\GraphicPrimitives\PG2NetworkHelper.dll** by the new one.

Short excurse into network transmission

The most usual low level networking protocols are **TCP** and **UDP**.

TCP

Imagine TCP as a phone call. One side (named the client) calls the other side (named the server). The server picks up the phone and now both server and client can send data (or talk to each other). As they are done, both sides hang up the phone.

TCP takes care about all data arrives the other side. There is an internal mechanism of acknowledgement, re-transmission, and serialization. If TCP fails to transmit data, it reports an error to the application and closes the connection. TCP transmissions are not limited in time or data size.

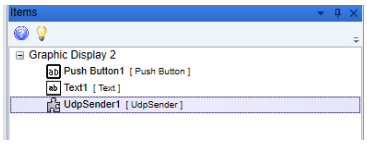
UDP

UDP as the opposite is like a SMS message. The sender just throws out some small data (or text) to a receiver. UDP itself does not take care about if a sent data package arrives the receiver or not. The size of single data packages is also very limited. Depending on your network configuration the maximal UDP payload is between 508 and 65507 Bytes.

Configuration

After successful install, two new controls appearing in your Process Graphics Editor's toolbox.

Both controls can be added to a graphics control, but because they are invisible, they cannot be selected on the graphic. You have to use the item list to select this controls.



TcpClient

This is a TCP client to connect to external TCP servers.

Properties:

<table>
<tr><td>Connect</td><td>Triggers the connection to the server<table>
<tr><td>True</td><td>Connect to the server, hold the connection and reconnect if the connection is lost</td></tr>
<tr><td>False</td><td>disconnect</td></tr>
</table></td></tr>
<tr><td>Encoding</td><td>Text in PG2 is stored as 16bit Unicode. This property sets, how the text is encoded before sent to the network, and how the received data is decoded. Note there is no codepage support except of the systemdefault.<table>
<tr><td>ASCII</td><td>ancient 7bit charset</td></tr>
<tr><td>BINARY</td><td>just use the lower 8bit on send, ASCII on receive</td></tr>
<tr><td>Systemdefault</td><td>Use the codepage used as system default</td></tr>
<tr><td>UTF*</td><td>Use UTF encoding</td></tr>
</table></td></tr>
<tr><td>Hostname</td><td>Hostname or IP of the TCP server</td></tr>
<tr><td>Hostport</td><td>Portnumber of the server</td></tr>
<tr><td>ResponseAppend</td><td>This is how received data will be stored to “ResponseData”<table>
<tr><td>AllwaysAppend</td><td>all received data will be stored</td></tr>
<tr><td>AllwaysOverwrite</td><td>only the last data block will be stored</td></tr>
<tr><td>ClearOnConnect</td><td>all data from this connection period will be stored</td></tr>
</table></td></tr>
<tr><td>ResponseData</td><td>String variable to store received data. Received Data will be dropped if this is null.</td></tr>
<tr><td>ResponseTimeout</td><td>How much Milliseconds the connection will be kept open for response data, if only “SendTrigger” is used.</td></tr>
<tr><td>SendData</td><td>Text to be sent. PG2 allows also regular expressions like \n, \t and \u0000.</td></tr>
<tr><td>SendTrigger</td><td>Send data on rising edge. If “connect” is false, a connection will be opened first, and closed after “ResponseTimeout”. The connection also stays open as long as “SendTrigger” is true.</td></tr>
<tr><td>Status</td><td>Integer variable to monitor the current connection state.<table>
<tr><td>0</td><td>Disconnected</td></tr>
<tr><td>1</td><td>currently trying to connect</td></tr>
<tr><td>2</td><td>Connected</td></tr>
</table></td></tr>
</table>

UdpSender

This can be used to send single UDP data packets to somewhere else.

Properties:

<table>
<tr><td>Encoding</td><td>Text in PG2 is stored as 16bit Unicode. This property sets, how the text is encoded before sent to the network. Note there is no codepage support except of the systemdefault.
<table>
<tr><td>ASCII</td><td>ancient 7bit charset</td></tr>
<tr><td>BINARY</td><td>just use the lower 8bit on send</td></tr>
<tr><td>Systemdefault</td><td>Use the codepage used as system default</td></tr>
<tr><td>UTF*</td><td>Use UTF encoding</td></tr>
</table></td></tr>
<tr><td>Hostname</td><td>Name or IP of the receiver. UDP also allows to use a broadcast IP address.</td></tr>
<tr><td>Hostport</td><td>Portnumber of the receiver.</td></tr>
<tr><td>SendData</td><td>Text to be sent. PG2 allows also regular expressions like \n, \t and \u0000.</td></tr>
<tr><td>SendTrigger</td><td>Send data on rising edge.</td></tr>
</table>

TcpServer and UdpReceiver

You may have noticed, there is no TcpServer and UdpReceiver control. The reason for this is, that you can have only one control listening to a specific port per IP address. So if you open the same graphic twice, only one would work. I also cannot imagine any serious use-case for this kind of controls.

Examples

Control security cam

We have some PTZ cam's who can be controlled by using the simple serial line protocol "Visca". This serial line is connected to a network converter, and that one is configured to dump every UDP packet on a specific port to the serial line.

The Cam moves to the stored position if I push the button. That's witchcraft, dudes!

On the PG2 side we have a Push Button, a trigger variable (DateTime), and a UdpSender. One per each camera function.

The Push button properties:

Target	trigger
Value	_Now + 1.

The UdpSender properties:

Encoding	BINARY
Hostname	172.16.8.234
Hostport	9458
SendData	"\u0081\u0001\u0004\u003f\u0002\u0001\u00ff"
SendTrigger	trigger > _Now

The text in SendData is the command code for "move to position 1". Note that the PG2 editor replaces some of the regular expression numbers by actual characters. This does not break the functionality, but makes further edition much harder.

Get a website

Well, this was just for testing purpose. But maybe you can find some use in this example.

The Textbox displays the source code of the webpage if I push the button.

We need a Push Button, a trigger variable (DateTime), a response variable (String), a TcpClient, and a Text display.

The Push Button Properties:

Target	trigger
Value	_Now + 1.

The TcpClient Properties:

Connect	false
Encoding	UTF8
Hostname	“webserver”
Hostport	80
ResponseAppend	ClearOnConnect
ResponseData	response
ResponseTimeout	1000
SendData	“GET /index.html\n\n”
SendTrigger	trigger > _Now
Status	null

The Text component properties:

Text	response

Get a Freelance AC800F buffer battery state, using a webserver as gateway

The textbox is green if the Battery is OK, and red if it's bad. Refreshes every 8 Seconds. Does an ugly flash, but who cares. Should I create something like this for production use?

We need a php-file on the webserver, a response variable (String), a TcpClient, and something to show a color (I use a text here too).

The php-file (batt.php):

```
<?php
if (isset($_REQUEST['ip']))
{
	$ip = str_replace(";","",$_REQUEST['ip']);
	$handle = @fopen("http://".$ip."/main.htm","r");
	$text = "";
	if ($handle)
	{
		while (!feof($handle))
		{
			$text = $text.fgets($handle);
		}
		$pos1a = strpos($text, "<div style=\"position:absolute;top:90px; left:346px\">")+52;
		$pos1b = strpos($text, "</div>", $pos1a);
		$pos2a = strpos($text, "<div style=\"position:absolute;top:90px; left:395px\">")+52;
		$pos2b = strpos($text, "</div>", $pos2a);
		if (substr($text, $pos1a, $pos1b-$pos1a) == "<img src=\"grn4.gif\">" && substr($text, $pos2a, $pos2b-$pos2a) == "<img src=\"grn4.gif\">")
		{
			echo "GOOD";
		}
		else
		{
			echo "BAD";
		}
	}
	else
	{
		echo "FAIL";
	}
}
else
{
	echo "FAIL";
}
 ?>
```

The TcpClient Properties:

Connect	false
Encoding	UTF8
Hostname	"webserver"
Hostport	80
ResponseAppend	ClearOnConnect
ResponseData	response
ResponseTimeout	1000
SendData	"GET /batt.php?ip=172.16.1.4\n\n"
SendTrigger	(_Now#Second & 7) = 0
Status	null

The Text properties:

FillColor	if response = "GOOD" then Green else if response = "BAD" then Red else Transparent

Write an entry to our custom production log

This example is maybe not useful for you, but it was the first reason for me to create the network helper controls. It was the last thing I had to convert from an old VB6 graphic.

I use a push button, a trigger variable (Boolean), a response variable (String), a TcpClient and a lot of external values.

Push button properties:

Enabled	!trigger
Target	trigger
Value	true

Yes, this button needs to be a one-shot thing!

TcpClient properties:

Connect	false
Encoding	BINARY
Hostname	ap01
Hostport	6543
ResponseAppend	ClearOnConnect
ResponseDate	response
ResponseTimeout	100
SendData	"ADD Isorechner2 " + FormatDateTime("g", "de-DE", _Now#ToLocal) + "\u0001TC-" + $'OP-Nummer' + "\u0001" + FormatReal(MMTMenge:KVal, 1) + "\u0001" + FormatReal(ISOGehalt:KVal, 1) + "\u0001" + FormatReal(Val, 1) + "\u0001-\u0001" + FormatReal(FQ42612_TOT:IN, 1) + "\u0001" + FormatReal(L52600:IN, 1) + "\u0001" + FormatReal(T52600:PV, 1) + "\u0001" + FormatReal(FQ63709:IN, 1) + "\u0001\u0001\u0001Eintrag Berechnung Isocyanat\u0001" + SchichtFarbe + "\u0001\u0000"
SendTrigger	trigger && response = “”
Status	null