Remote Panel Development

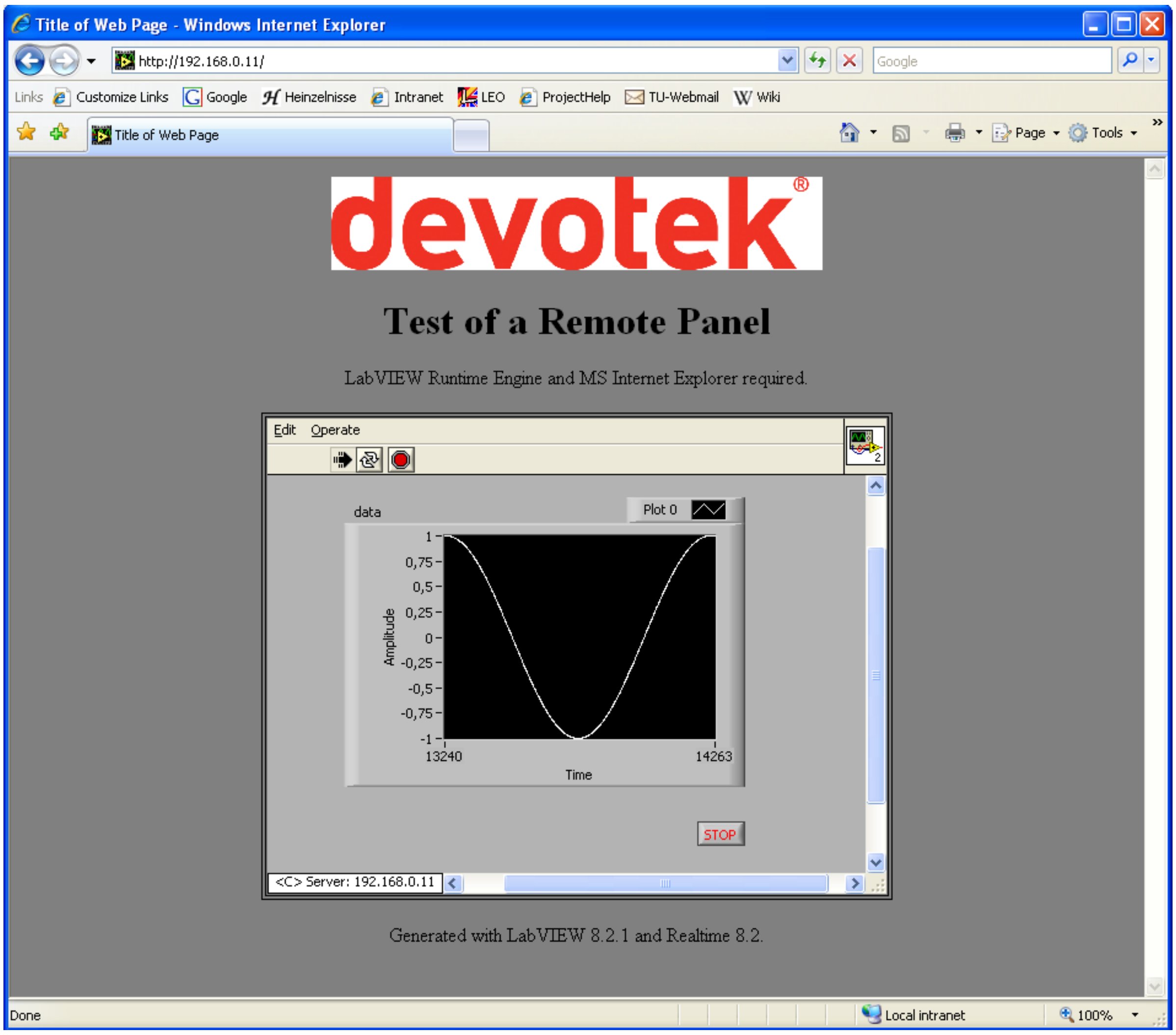
with LabVIEW Realtime

Mathias Lindner

September 25, 2007

1 Checklist for developing real time applications with a remote panel

1. Build your VIs on the real time target as usual. Instead of building the user interface on the real time host, implement it right on the real time target.
2. Right-click the real time target, chose Properties → Web Server: Configuration and check the box at Enable Web Server.
3. Deploy and run your VI.
4. Open Tools → Web Publishing Tool. . . . Choose your VI with the user interface from the drop-down list. Select Embedded Viewing Mode and tick the box Request control when connection is established. When required change the document title, the header and the footer of the later website on the next page. This could be edited later too. After that, type in a local directory to save the HTML-file. Click Save to Disk and confirm the hint, that The selected path is not under the Web Server root directory. Your website is now built.
5. If you want to change the design of your website, your free to edit the HTML-file like any other HTML-document now. For further information go to section 3.
6. Rename the final document to *index.htm*. It's important to choose the extension HTM.


Title of Web Page - Windows Internet Explorer

http://192.168.0.11/

Test of a Remote Panel

LabVIEW Runtime Engine and MS Internet Explorer required.

<C> Server: 192.168.0.11

Generated with LabVIEW 8.2.1 and Realtime 8.2.

Figure 1: Example of a Remote Panel

7. Open the Measurement & Automation Explorer → Remote Systems and right-click your real time target. Choose File Transfer and log in anonymously. Skip to */ni-rt/system/www/* in the Remote Directory and to the previously specified folder in the Local Directory. Mark the *index.htm* and all other added files (as graphics, logos, . . .) and click To Remote ↑. Close the window after the successful transfer.

8. Verify that your VI is running on the real time target.

9. Make sure that your *Internet Explorer* do not use a proxy server when connecting with the real time target and that the real time target has a static ip-address.

10. Type the following address in your *Internet Explorer*: http://<real time target ip> (e. g. http://192.168.0.11). This should load your website with the running VI plugged in – an example is given in Figure 1.

2 Network traffic

The VI in Figure 1 was tested. It generates a sine function and displays it in a waveform chart. The sine is built from dots – one each 10 ms. One remote panel was opened in controlling mode. The traffic direction is seen from the real time target. Thus, upload means traffic from the target to the remote panel, download describes traffic from the remote panel to the real time target.

Remote panel update rate	Upload rate	Download rate
10 ms	8.3 kB/s	2.1 kB/s
1000 ms	1.7 kB/s	54 B/s

In the first test the chart was drawn always, when a new dot was calculated. The update rate of the remote panel was 10 ms consequently.

Secondly, the update rate was limited to 1 s. Therefore, the calculated dots were stored into a FIFO. They were read and displayed in the chart only once a second. To guarantee a fluent user interaction the input elements (like the stop button) are still scanned every 10 ms.

3 HTML-Code

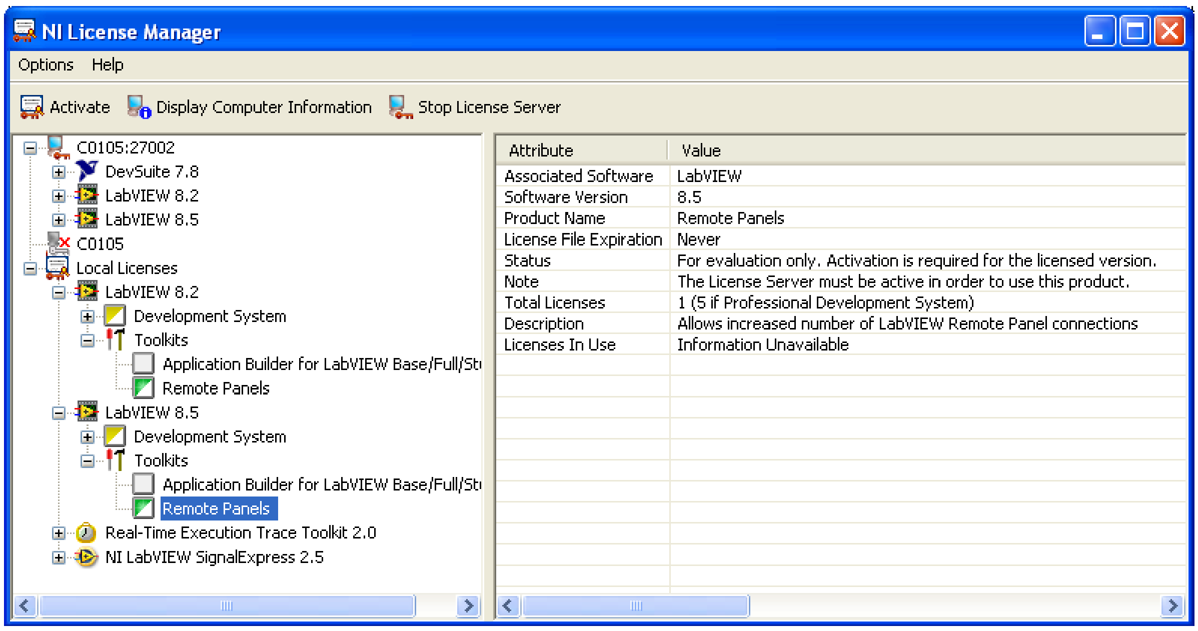
```
<OBJECT ID="LabVIEWControl" CLASSID="CLSID:A40B0AD4-B50E-4E58
   -8A1D-8544233807AF" WIDTH=529 HEIGHT=495 CODEBASE="ftp://ftp
   .ni.com/support/labview/runtime/windows/8.5/LVRunTimeEng.exe
   ">
<PARAM name="LVFPPVINAME" value="Test.lvproj/LEA1/target.vi">
<PARAM name="REQCTRL" value=true>
<EMBED SRC=".LV_FrontPanelProtocol.rpvi85" LVFPPVINAME="Test.
   lvproj/LEA1/target.vi" REQCTRL=true TYPE="application/x-
   labviewrpvi85" WIDTH=529 HEIGHT=495  PLUGINSPAGE="http://
   digital.ni.com/express.nsf/bycode/exck2m">
</EMBED>
</OBJECT>
```

This is the only important code, which needs to be included into the HTML-file. It displays a plugin-frame with the VI *target.vi* of the project *Test.lvproj*. If the *LabVIEW Runtime Engine* is not installed on the remote PC, the browser will ask you to install it from `ftp://ftp.ni.com/support/labview/runtime/windows/8.5/LVRunTimeEng.exe`. This address could be changed to another one, e.g. on a distribution server.

The rest of the file is free to design. You can add pictures, links, text etc.

4 License Management

For every user connecting to the website one license is needed. The amount of available licenses could be seen in the *NI License Manager* (open it from the NI program folder in the Windows start menu). Compare it to Figure 2.



```
C0105:27002
    DevSuite 7.8
    LabVIEW 8.2
    LabVIEW 8.5
C0105
Local Licenses
    LabVIEW 8.2
        Development System
        Toolkits
            Application Builder for LabVIEW Base/Full/St
            Remote Panels
    LabVIEW 8.5
        Development System
        Toolkits
            Application Builder for LabVIEW Base/Full/St
            Remote Panels
    Real-Time Execution Trace Toolkit 2.0
    NI LabVIEW SignalExpress 2.5
```

Attribute	Value
Associated Software	LabVIEW
Software Version	8.5
Product Name	Remote Panels
License File Expiration	Never
Status	For evaluation only. Activation is required for the licensed version.
Note	The License Server must be active in order to use this product.
Total Licenses	1 (5 if Professional Development System)
Description	Allows increased number of LabVIEW Remote Panel connections
Licenses In Use	Information Unavailable

Figure 2: NI License Manager

The available licenses are written on the right side next to Total Licenses.

The *LabVIEW Full Development System* has one license included, the *LabVIEW Professional Development System* contains five. It is possible to extend this limitation by purchasing more licenses[1].

Article Number	Description	Costs
778562-05	LabVIEW Remote Panel Servers License – 5 Users	NOK 2,400.-
778562-20	LabVIEW Remote Panel Servers License – 20 Users	NOK 9,600.-
778562-50	LabVIEW Remote Panel Servers License – 50 Users	NOK 23,600.-

1 http://sine.ni.com/nips/cds/view/p/lang/en/nid/11017

5 Several instances of the VI

As described in section 4 multiple clients are allowed to open the remote panel of the same VI simultaneously. By default, one of those users can control the VI whereas the other clients only have a reading access.

Since *LabVIEW 8.20* it is also possible to accept more than one controlling instance of the remote panel.[2] Therefore you have to bring your VI to the front, chose File → VI Properties and select Execution from the drop-down list. Then tick the option Reentrant execution. After deploying your VI to the real time target once again it should be possible to open more clients with control permission.

If more than one client is allowed to control a VI, more than one instance of the front panel would be opened. The clients can control the VI without being affected by the other clients.

6 Functionality Not Supported in Viewing and Controlling Remote Front Panels

The following list includes functionality not supported and recommendations to consider when viewing and controlling remote front panels[3].

- Because of the constraints of a Web browser, user interface applications that attempt to manipulate the dimensions and location of a front panel do not work properly when that front panel is displayed as a part of a Web page. Although the Web Server and the LabVIEW browser plug-in attempt to preserve the fidelity of complex user interface applications – in particular, those that present dialog boxes and subVI windows – some applications might not work properly in the context of a Web browser. National Instruments recommends that you do not export these types of applications for use in a Web browser.
- Avoid exporting VIs that have While Loops but no wait function. These VIs prevent background tasks from performing in a reasonable amount of time, making front panels unresponsive when viewed or controlled remotely.

2 http://zone.ni.com/devzone/cda/tut/p/id/4867
3 http://zone.ni.com/reference/en-XX/help/371361B-01/lvhowto/fnclty_nt_spprtd_rmt_pnl/

- Some VIs might not work exactly the same way from a remote computer as they do when run locally. .NET and ActiveX controls embedded on a front panel do not display on a remote client because they draw and operate almost completely independently of LabVIEW. If a VI presents the standard file dialog box, the controller receives an error because you cannot browse a file system remotely. Also, the browse button of a path control is disabled in remote panels.
- Clients viewing a front panel remotely might see different behavior depending on whether the front panel they are connecting to is from a built application. Specifically, if the front panel is from a built application, any programmatic changes to the front panel made before the client connects to the front panel are not reflected on the client computer. For example, if a Property Node changes a caption on a control before a client connects to that front panel, the client will see the original caption of the control, not the changed caption.
- Only a controller can remotely view the front panel of a VI dynamically opened and run using the VI Server or the front panel of a subVI configured to display the front panel when called. Clients not controlling the VI cannot view the front panel.
- Block diagrams that achieve certain user interface effects by polling properties of a front panel control might experience decreases in performance when you control the VI from a remote computer. You can improve the performance of these VIs by using the Wait for Front Panel Activity function.
- LabVIEW cannot generate a Panel Close event for a VI being viewed or controlled remotely. If you are viewing or controlling a VI remotely, LabVIEW can generate events only in the Control class, that is not in the Application or VI class.

Another common component of most applications is a system dialog box. System dialog boxes are a good way to notify a user of a problem, communicate important results, or solicit input from the user. The Simple Error Handler VI even makes use of the system dialog box to manage errors. System dialog boxes are great tools, but they do not work well with distributed applications. The system dialog box remains on the local computer instead of being distributed with the application. In the following example, the VI is simple. It uses an Event structure to monitor the status of a Boolean control. When the user clicks a button, the VI displays a one-button system dialog box with the message Button Pressed.

If you distribute this VI remotely and a user on the Web clicks the button, a dialog box appears on the host computer only. Because that dialog box is a local system box, it does not appear to the user on the Web, but it prevents anyone from using the VI until it is closed. So the user on the Web cannot use the VI but is unaware that the problem is the open dialog box. Because of the localized nature of ActiveX and system dialog boxes, you should not use these technologies in applications designed to be viewed and controlled remotely.[4]

Summery[5]:

- Property nodes that manipulate the location and dimensions of the front panel my not work properly.
- Avoid while loops with no wait functions.
- ActiveX controls embedded on a front panel do not display on a remote client.
- You cannot use the standard file dialog box.
- The file path browse button is disabled.
- If the front panel is from a built application, any programmatic changes to the front panel made before the client connects to the front panel are not reflected on the client computer.
- Only a controller can remotely view the front panel of a VI dynamically opened and run using the VI Server or the front panel of a subVI configured to display the front panel when called. Clients not controlling the VI cannot view the front panel.
- Polling properties of a front panel control might experience decreases in performance.

4 http://zone.ni.com/devzone/cda/tut/p/id/3277#toc3

5 http://digital.ni.com/public.nsf/allkb/21B9A9C913F54F918625703D004D6FD9