TEMPERATURE CHAMBER QUICK START GUIDE

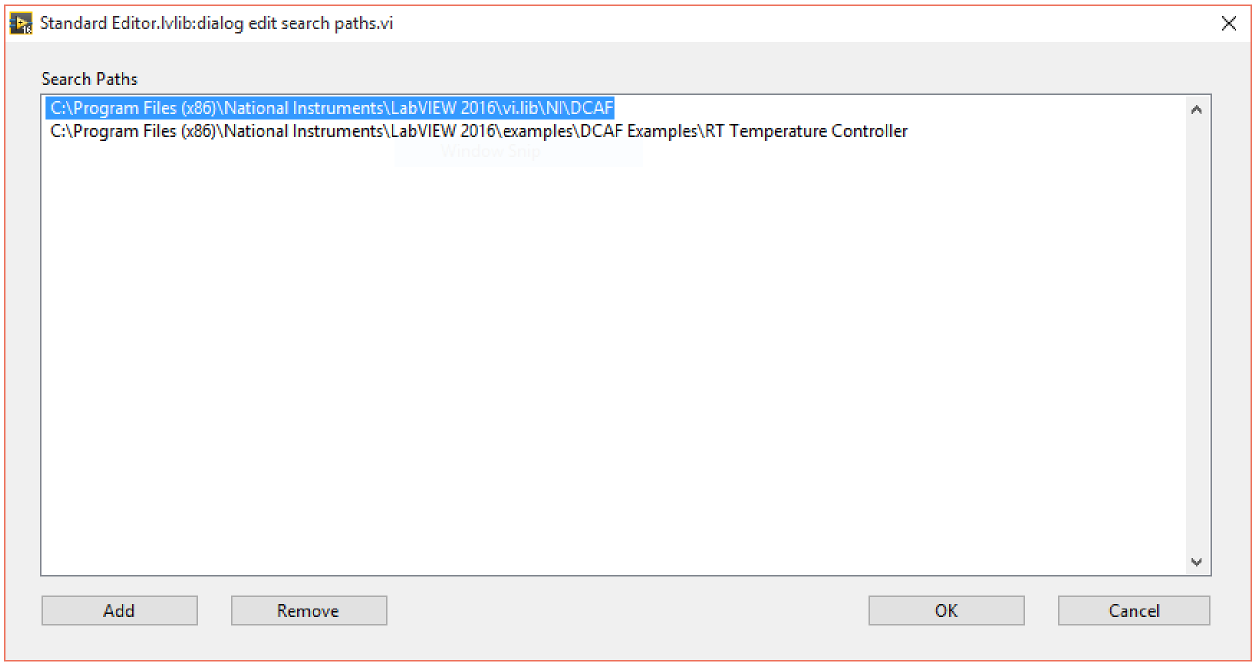
Introduction

This example demonstrates the implementation of a simple temperature chamber controller application using the Distributed Control and Automation Framework (DCAF). This example makes use of a model of the chamber to simulate its I/O and allows users to define the setpoint and PID gains of the control algorithm through a simple user interface. This document describes the minimum number of steps required to get this example running. For more details on how to navigate the framework and customize the example, see the 'Temperature Chamber Detailed Documentation'.

This example can either be run exclusively on a Windows machine, or it can be run such that the UI runs on Windows while the control logic runs on a controller with a real-time OS. If running on Windows only, follow the instruction steps marked <WINDOWS ONLY>. If running on two targets, follow the instruction steps marked <WINDOWS and cRIO>.

Execution Steps

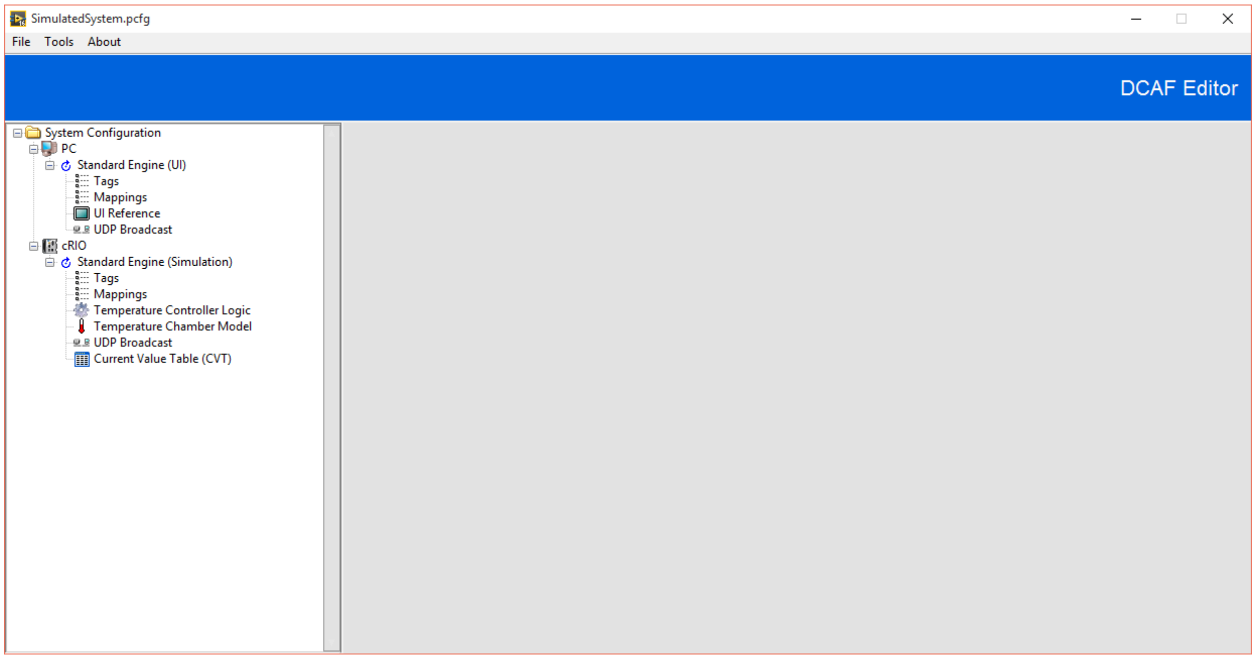
1. Open up the Standard Configuration Editor for DCAF by navigating in LabVIEW to **Tools>>DCAF>>Launch Standard Configuration Editor…**
2. Navigate within the editor to **Tools>>Edit Plugin Search Paths**.
3. Add a search path to the DCAF plugins for this example located at **<LabVIEW examples>\DCAF Examples\RT Temperature Controller** if it's not already there. Also confirm that the standard vi.lib file paths are specified as shown below for the version of LabVIEW that you are using.


Standard Editor.lvlib:dialog edit search paths.vi

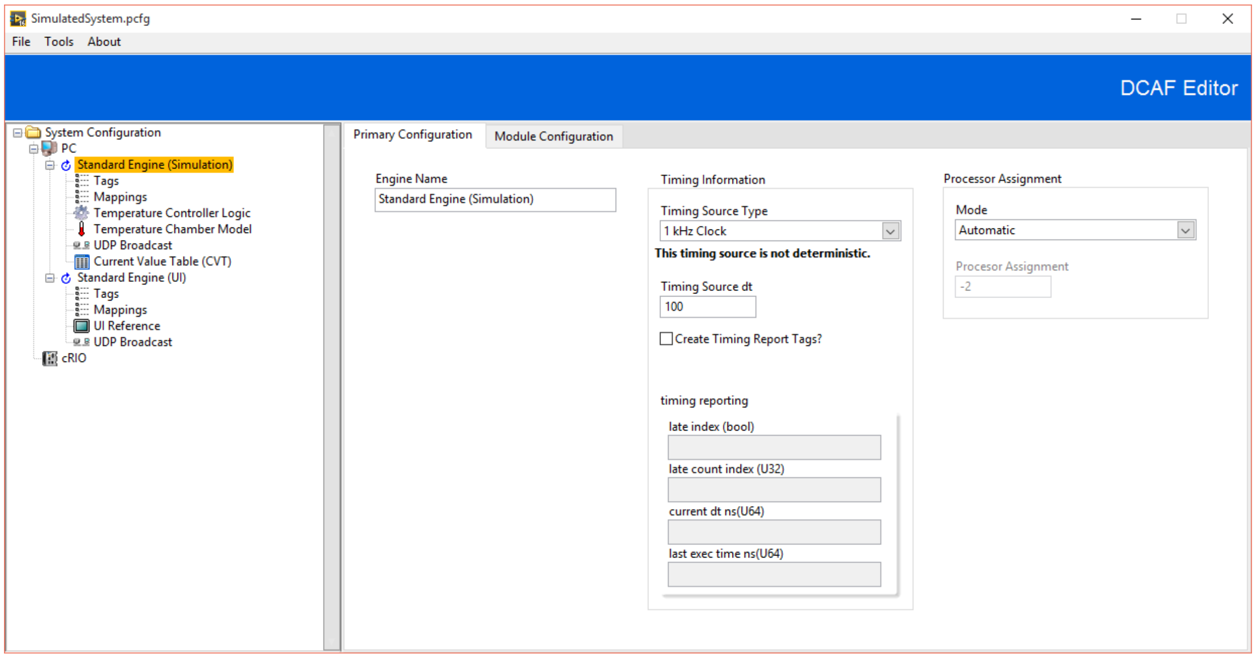
Search Paths

```
C:\Program Files (x86)\National Instruments\LabVIEW 2016\vi.lib\NI\DCAF
C:\Program Files (x86)\National Instruments\LabVIEW 2016\examples\DCAF Examples\RT Temperature Controller
```

4. Click **OK** to confirm the new search path. The configuration editor will now scan these directories for any DCAF plugins and load them into memory.
5. Once the busy cursor disappears, navigate to **File>>Open** and open up the configuration file for this example at **<LabVIEW Examples>\DCAF Examples\RT Temperature Controller\SimulatedSystem.pcfg**.



6. <WINDOWS ONLY> If running this configuration purely on your local Windows machine, modify the configuration so that the Engine on the cRIO runs on the PC instead. To do this, click the **Standard Engine (Simulation)** node under the engine and drag it up to the PC.



Engine Name: Standard Engine (Simulation)

Timing Source Type: 1 kHz Clock

This timing source is not deterministic.

Timing Source dt: 100

Create Timing Report Tags? ☐

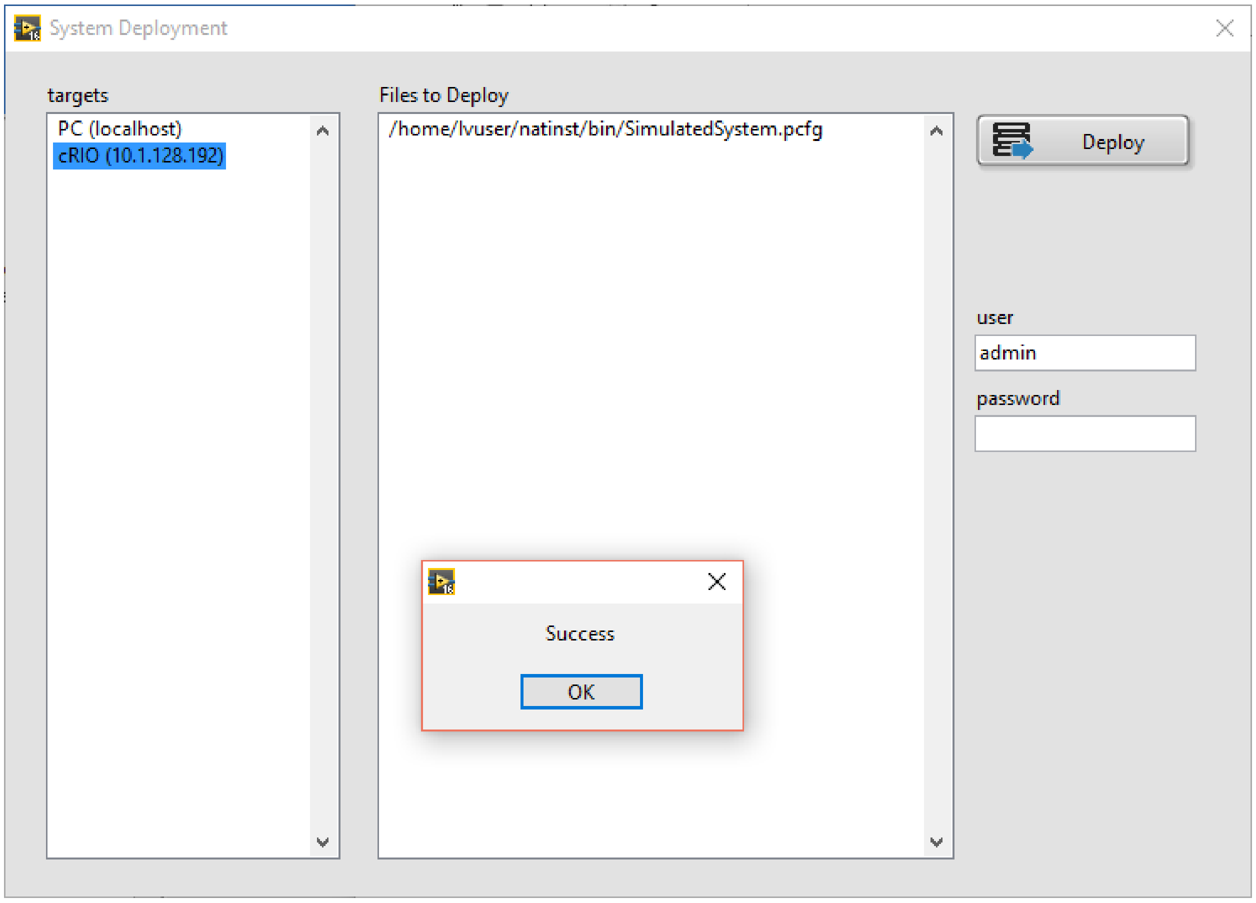
Mode: Automatic

Procesor Assignment: -2

7. Ensure proper IP addresses for both UDP modules.
   a. <WINDOWS ONLY> For the UDP module under each engine, set the **send to address** field on the **Module Settings** tab for both UDP modules to **localhost**.
   b. <WINDOWS and cRIO> For the UDP module under each engine, set the **send to address** on the **Module Settings** tab on the PC UDP module to the IP Address of the cRIO and vice versa

for the cRIO UDP module. Be sure to specify the actual IP address and not use a value of **localhost**.

8. Save your changes to the configuration file by navigating within the editor to **File>>Save**.
9. <WINDOWS and cRIO> Click the cRIO target in the configuration editor and enter its IP address in the **IP** field. Then navigate to **Tools>>Deploy Tool**, click the cRIO target in the targets list, and then click the **Deploy** button. Ensure that you see a Success dialog appear as shown below.



System Deployment

targets

PC (localhost)

cRIO (10.1.128.192)

Files to Deploy

/home/lvuser/natinst/bin/SimulatedSystem.pcfg

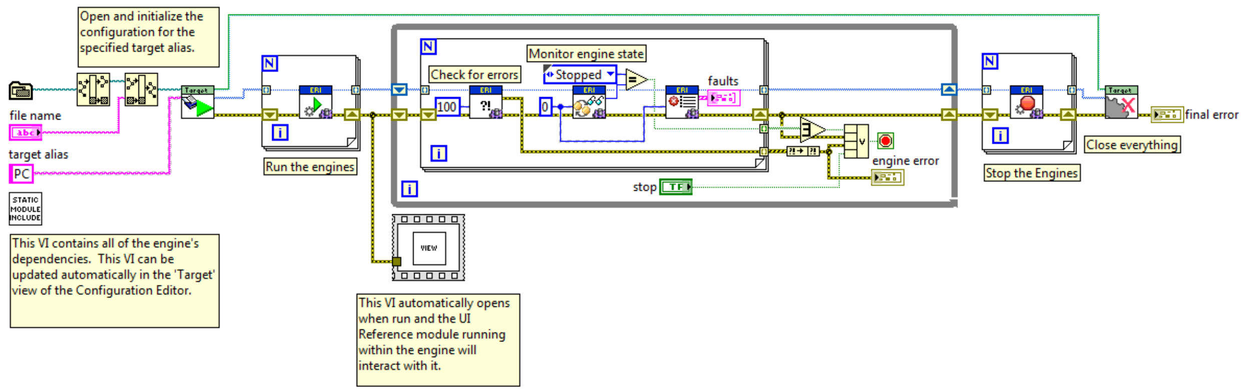
user

admin

password

Success

10. Open the **<LabVIEW Examples>\DCAF Examples\RT Temperature Controller\Runtime\Temperature Controller Example.lvproj** if it isn't already open. Then open the **Host Main.vi** from the project and inspect its Block Diagram.

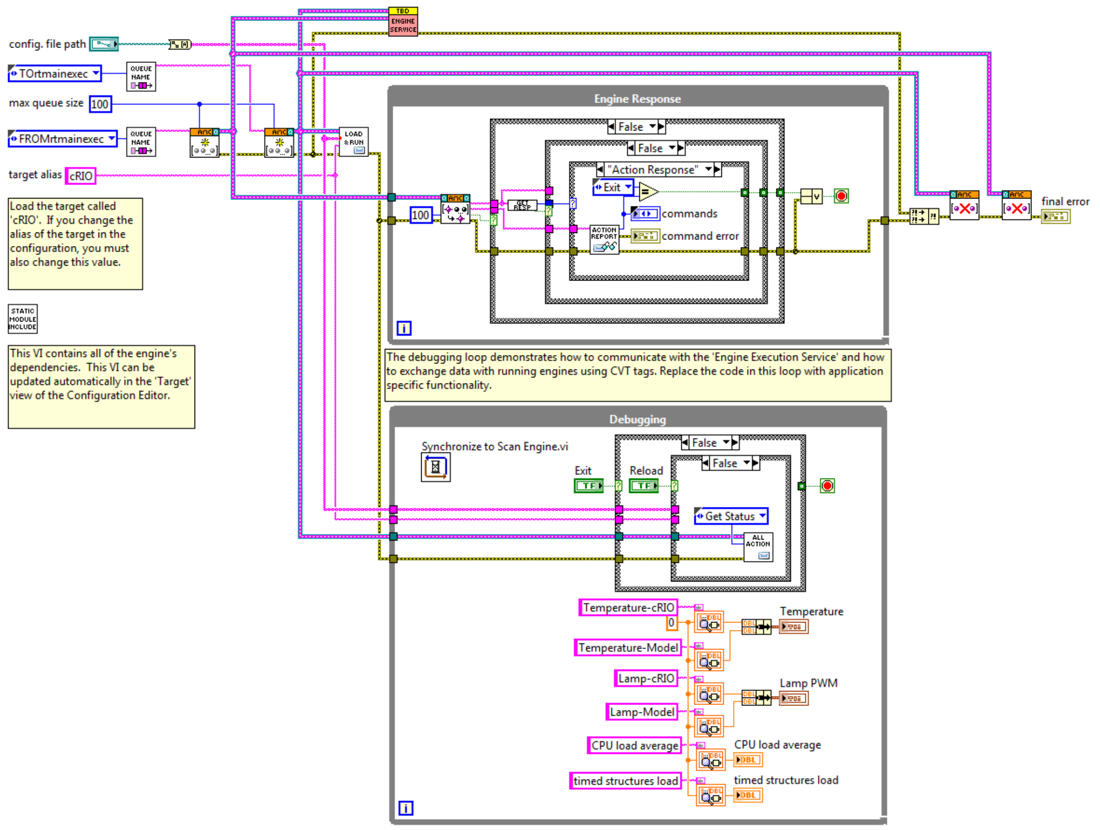


Open and initialize the configuration for the specified target alias.

This VI contains all of the engine's dependencies. This VI can be updated automatically in the 'Target' view of the Configuration Editor.

This VI automatically opens when run and the UI Reference module running within the engine will interact with it.

11. Run 'Host Main.vi'.
12. While still running 'Host Main.vi', open up and inspect 'cRIO Main.vi' in the project.



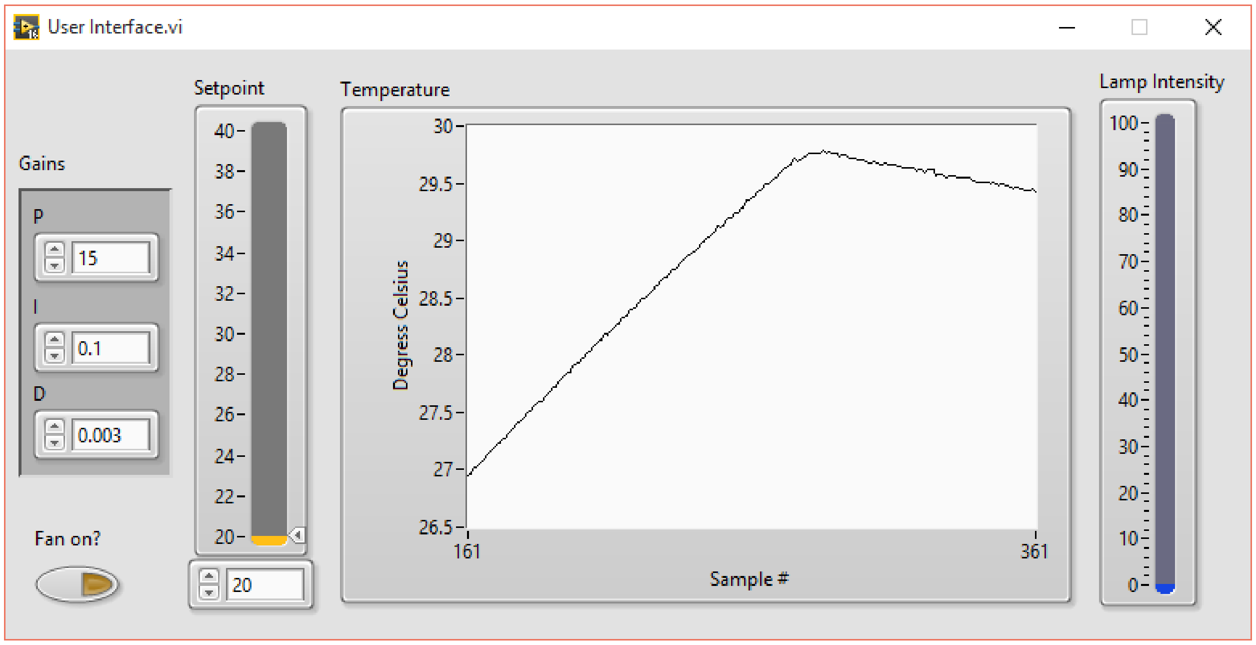
Load the target called 'cRIO'. If you change the alias of the target in the configuration, you must also change this value.

This VI contains all of the engine's dependencies. This VI can be updated automatically in the 'Target' view of the Configuration Editor.

The debugging loop demonstrates how to communicate with the 'Engine Execution Service' and how to exchange data with running engines using CVT tags. Replace the code in this loop with application specific functionality.

13. <WINDOWS and cRIO> Change the cRIO target's IP address in the project to match the IP of the controller you want to use.
14. <WINDOWS and cRIO> Run 'cRIO Main.vi'.

Once running, you should now see data appear on the ‘Host Main.vi’. (If data doesn’t appear, make sure that your IP settings for the UDP module are specified properly in the system configuration and that your firewall isn’t blocking the UDP communication.)


User Interface.vi
Gains
P
15
I
0.1
D
0.003
Fan on?
Setpoint
40
38
36
34
32
30
28
26
24
22
20
20
Temperature
Degress Celsius
30
29.5
29
28.5
28
27.5
27
26.5
161
361
Sample #
Lamp Intensity
100
90
80
70
60
50
40
30
20
10
0