Bergmans Mechatronics LLC

LabSocket

User Guide

LabSocket v3.5.2.7

May 2024

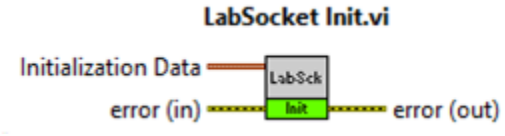
BML Document BML-2024-105.5.3

Table of Contents

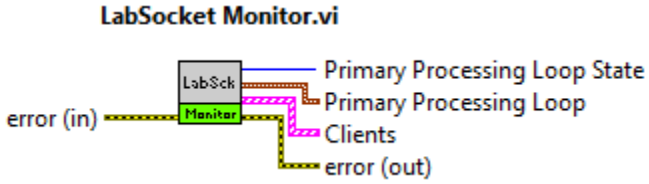
Version Notes

LabSocket version 3.5.2.7 (released May 2024) includes the following new capabilities relative to version 3.4.1.77 (released Jun 2015):

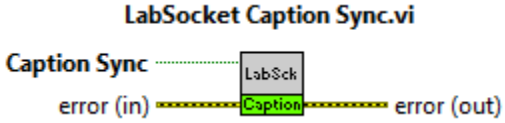
- Table support
- ColorBox support
- Mouse and touch support for cursors in XY graphs and XY graph curve visibility control using legend
- Mouse and touch support for sliders as images (including pop-up display or current value and increment coercion)
- Synchronization of units for numeric controls and indicators
- Support for inserting dynamically updated animated GIFs into browser
- Support for coercion of front panel numeric controls (eg. minimum, maximum and increment values)
- Support for password style in text controls and indicators
- System parameter initialization, using new LabSocket Init VI



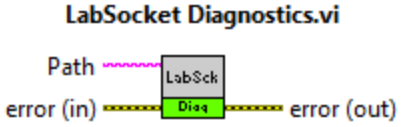
- System monitoring, using new LabSocket Monitor VI



- Caption synchronization, using new LabSocket Caption Sync VI



- Detailed diagnostics data output, using new LabSocket Diagnostics VI



1. Introduction

LabSocket enables remote access to LabVIEW applications from desktop or mobile web browsers without the need for browser plugins or a client-side run time engine. The system automatically creates a web page representation of a LabVIEW application ("Target VI") front panel and dynamically synchronizes the contents of the web page and front panel.

A key technology used in the system is the HTML5 WebSocket interface that enables continuous, bi-directional communication with a web browser. The use of this interface contributes to the name of the system: "LabSocket = LabVIEW + WebSocket".

This User Guide describes the setup and operation of the LabSocket software. The guide assumes that the user is familiar with LabVIEW but has only a basic level of Linux and server virtualization experience.

1.1 Supported Platforms

LabSocket operates on Windows PC (Windows XP or later) and NI cRIO-903x Real-Time controllers. On cRIO-903x platforms, the system only operates in "Headless" operation mode with the embedded UI on.

For browser access to either cRIO-903x controllers in other configurations or to other Real-Time controllers, consider the use of LabSocket-Embedded (http://labsocket.com/LabSocket-E.html).

1.2 Client Mapping

On Windows platforms, LabSocket can be configured for one of two client mapping configurations: Basic mode supports 1:N Target VI-to-browser client mapping. In this mode one or more browsers connect to the same instance of a Target VI.

MultiClient mode supports N:N Target VI instance-to-browser client mapping. In this mode, each browser that connects to the system is mapped to a unique instance of the Target VI.

On CompactRIO 903x platforms, only Basic mapping is currently supported.

The supported mappings are summarized in Table 1.1

Table 1.1 Client Mapping Modes

	Client Mapping Mode	
Platform	**Basic**	**MultiClient**
Desktop	**Yes**	**Yes**
cRIO-903x	**Yes**	No

Mapping modes are discussed in detail in Section 3, *Theory of Operation*.

1.3 Downloading and Activating LabSocket VIPM File

LabSocket is available for download via the LabVIEW VI Package Manager or the LabVIEW Tools Network (Section 4.2, *LabSocket Software Installation*). The LabSocket VIPM package file may be used in evaluation mode for up to 30 days.

To activate LabSocket, purchase a license from the LabVIEW Tools Network (www.ni.com/labviewtools/labsocket) or from http://labsocket.com/purchase.html. Then in LabVIEW, select "Help > Activate Add-ons" and enter the license ID and password provided via e-mail after purchase.

1.4 LabSocket Server

An important component of the system is the "LabSocket Server" software that acts as a bridge between LabVIEW and the browser. This software consists primarily of an HTTP server and message broker. The two main implementations, of this software are:

1. a customer-operated Virtual Machine (most popular for production); and,
2. an existing cloud server (for getting started quickly with the system).

A notable benefit of the VM implementation is that it enables the entire LabSocket system to be easily set up within a customer LAN. Additional details about the download, setup and operation of the LabSocket Server VM are provided in Section 4.1, *LabSocket Server Virtual Machine Installation*.

LabSocket Server VM is available at no cost.

1.5 Licensing, Executables and Real-Time Deployments

The LabSocket licensing agreement allows the software to be installed on a single developer platform. From this single development platform, developers may:

a) create an unlimited number of standalone Windows executables that incorporate the LabSocket software and deploy these executables to an unlimited number of Windows platforms; and,

b) create an unlimited number of Real-Time applications that incorporate the LabSocket software and deploy these applications to an unlimited number of CompactRIO 903x devices.

Full licensing terms can be found under "Help > LabSocket License..." after LabSocket installation. See Section 6 for instructions on creating executables and real-time applications.

1.6 Contact

For assistance with system setup or questions about the system, please contact:

John Bergmans
Bergmans Mechatronics, LLC
e-mail: jbergmans@bergmans.com

2. System Requirements

2.1 Browsers

Browsers that use the LabSocket system must be compatible with the HTML5 WebSocket interface standard. Fortunately, most modern mobile and desktop browsers are compatible with this standard. A complete list of WebSocket-compatible browsers is shown at: http://caniuse.com/websockets

For users who require support for older browsers such as Internet Explorer 9.0 and earlier, a commercial gateway product can be provided to emulate the behavior of the WebSocket interface.

2.2 LabVIEW Development Platform

LabVIEW	LabVIEW 2014 32-bit Full Development System or later. Support for earlier versions of LabVIEW may be available upon request.
Minimum Hardware	2.3 GHz CPU and 4 GB memory
Operating System	Windows XP or later

2.3 LabSocket Server Virtual Machine Host

The LabSocket Server Virtual Machine (VM) operates within the free Oracle VirtualBox virtualization application. Platform requirements for the LabSocket Server Virtual Machine (VM) host are

Minimum Hardware	2.3 GHz CPU and 4 GB memory
Operating Systems	Windows, Mac OS X, Linux, Solaris

Additional details about VirtualBox host platforms can be found at: https://www.virtualbox.org/manual/ch01.html - hostossupport

Note that it is feasible to install VirtualBox and the VM on the LabVIEW Host Platform.

The LabSocket Server VM can also be supplied as a Parallels VM upon request.

2.4 Supported NI Real-Time Linux Platforms

LabSocket is compatible with CompactRIO cRIO-903x series NI Linux Real-Time controllers in headless mode and with the embedded UI turned on.

For browser access to either cRIO-903x controllers in other configurations or to other Real-Time controllers, consider the use of LabSocket-Embedded (labsocket.com/LabSocket-E.html).

2.5 LabSocket Server Ports

LabSocket Server ports 80 and 61614 must be accessible to the browser and LabSocket Server port 61613 must accessible to the LabVIEW Host Platform. The system may be modified to enable all browser traffic to use only port 80. Contact Bergmans Mechatronics for details.

3. Theory of Operation

3.1 Overview

To enable remote access to a LabVIEW application, or "Target VI", over the Web, LabSocket operates "LabSocket Support VIs" on the LabVIEW Host Platform in parallel to the Target VI. The LabSocket Support VIs operate unobtrusively and perform two key functions:

1. Upon starting the system, the Support VIs perform a screenscrape of the Target VI front panel. This function involves generating a web page using HTML and JavaScript code representing the Target VI front panel. This code is then posted to an HTTP server on the LabSocket Server platform.

2. After performing the screen scrape, the Support VIs continuously synchronize the properties of the elements on the front panel of the Target VI with those in the browser.

3.2 Client Mapping Modes

LabSocket can operate in two Client Mapping Modes. Basic Client Mapping enables 1:N Target VI-to-browser client mapping. In this mode, each browser that connects to the system is mapped to the same Target VI (Figure 3.1). As a result, each remote user sees the same data and can also control the Target VI.

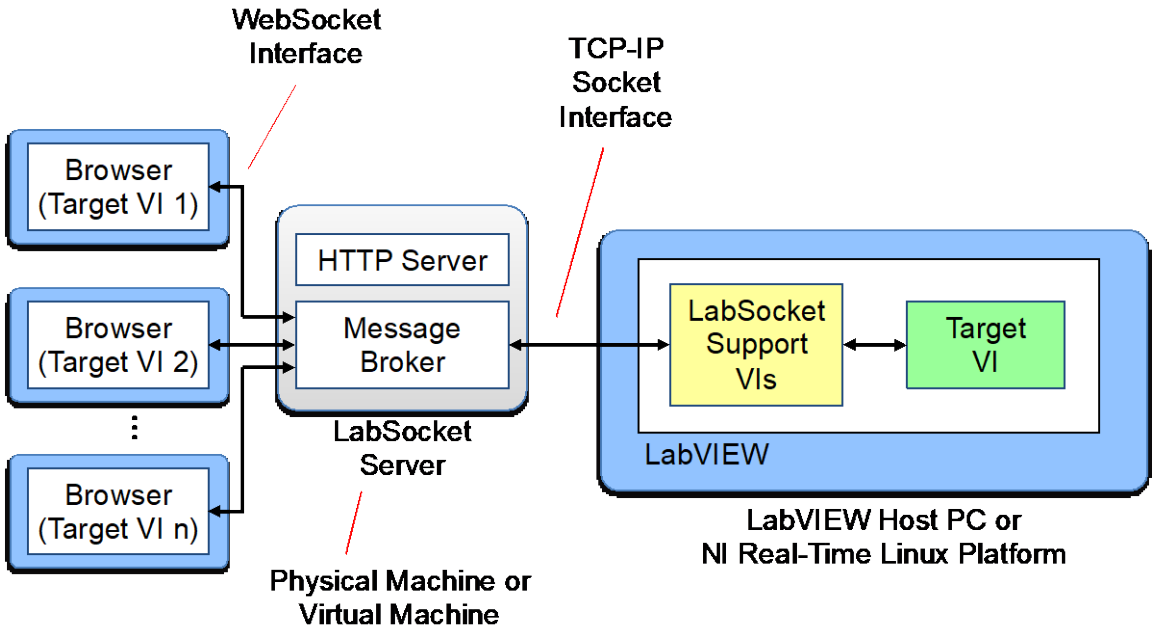


Figure 3.1. LabSocket in Basic Client Mapping Mode

MultiClient Client Mapping Mode supports N:N Target VI instance-to-browser client mapping. In this mode, each browser that connects to the system is mapped to a unique instance of the Target VI (Figure 3.2). MultiClient Mode is ideal for applications in which each user must operate the Target VI independently. Examples of such applications include database access systems and on-line quizzes.

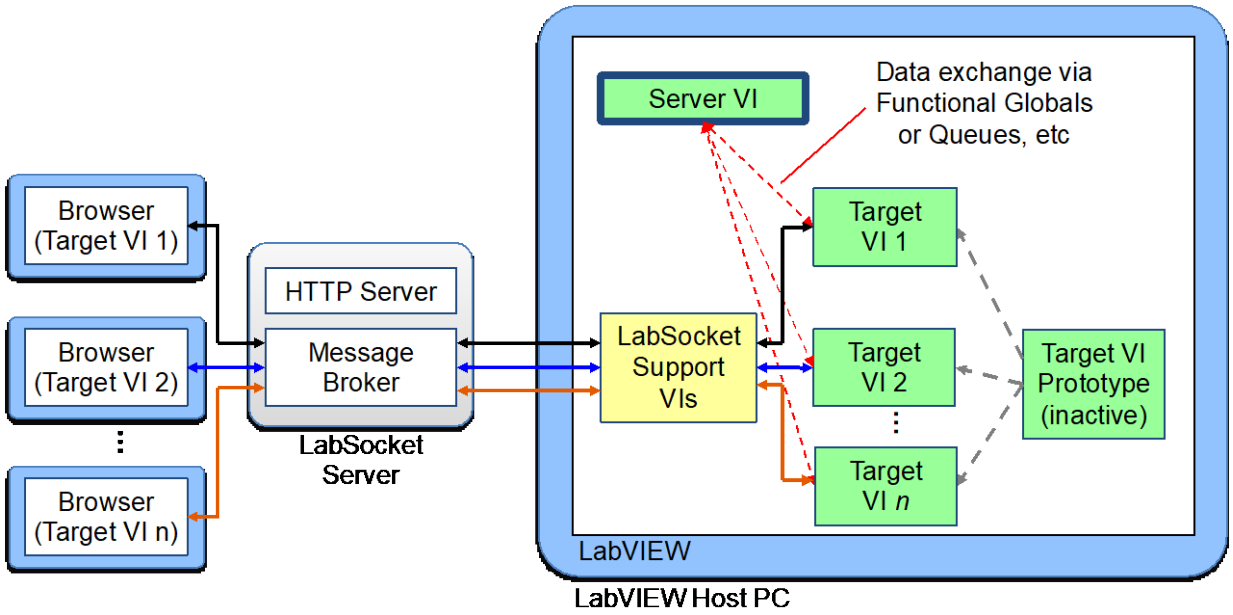


Figure 3.2. LabSocket System in MultiClient Client Mapping Mode

The Client Mapping mode is selected based on which LabSocket Start VI is added to the block diagram of the Target VI. "LabSocket Start.vi" and "LabSocket-MC Start.vi" invoke Basic and MultiClient mapping, respectively. Additional information on the use of these VIs is presented in Section 5, *LabSocket Application Development*. Wiring input details for the VIs is available in Section 8, *LabSocket Start VIs*.

3.3 LabSocket Server

The LabSocket Server software is a set of three programs that form a bridge between the browser client(s) and LabVIEW software: i) an Apache ActiveMQ message broker; ii) an Apache HTTP server; and, iii) a lightweight administrative program. The role of each component in this group is as follows:

1. The ActiveMQ message broker (http://activemq.apache.org/) transfers data between the LabSocket Support VIs and the browser. A key feature of this message broker is that it is able to establish a WebSocket interface to the browser. This interface enables continuous, bidirectional communications with the browser.

Another benefit of the use of the ActiveMQ message broker is that it includes a Lightweight Directory Access Protocol (LDAP) interface that enables the implementation of an optional LDAP-based user authorization mechanism (See Section 11, *User Administration*).

2. The HTTP server transmits HTML and JavaScript code to each browser that connects to the system.

3. Administrative software on the LabSocket Server platform manages the browser code that is received from the LabSocket Support Vis.

The two main physical implementations of the LabSocket Server software are:

1. A customer-operated Virtual Machine (most popular for production). A pre-configured VirtualBox Virtual Machine (VM) is available for use with evaluation and commercial versions of the LabSocket software.

   The use of a VM enables the entire LabSocket system to be easily set up within a customer LAN and assures customers complete control and ownership of the system and their data. The VM may be downloaded from the National Instruments LabVIEW Tools Network repository or the LabSocket website (See Section 4.1, *LabSocket Server Virtual Machine Installation*)

2. An existing Cloud Server (Quick start option) - This option is available for free for use with the LabSocket evaluation software and enables developers to get started quickly with the use of the system.

The use of both of these configurations of the system is described in this document.

Commercial users can also use one of the following two LabSocket Server implementations. Please contact Bergmans Mechatronics to discuss and request these options.

1. Customer Cloud Server - A pre-configured cloud server instance that contains the LabSocket Server software can be leased for the exclusive use of a customer.

2. Native Operation – Detailed instructions can be provided for setting up the individual LabSocket Server components directly on a customer's Windows, Mac or Linux platform.

4. Setup Instructions

This section provides step-by-step instructions on the installation of the LabSocket system. The instructions consist of two parts: i) LabSocket Server Virtual Machine installation; and, ii) LabSocket software installation.

Instructions to optionally uninstall the LabSocket software are also presented in this section.

4.1 LabSocket Server Virtual Machine Installation

NOTE - If you plan to use the LabSocket Server software on Bergmans Mechatronics' existing Cloud Server, proceed to Section 4.2, *LabSocket Software Installation*, to begin installation of the LabSocket software.

1. Download and install the Oracle VirtualBox virtualization application (http://virtualbox.org) onto the physical machine that will act as the host platform for the Virtual Machine (VM). This host can use a Windows, Mac OSX or Linux operating system and may be either the LabVIEW Host PC or a separate physical machine.

2. Download the LabSocket Server VM Open Virtualization Format Archive (OVA) file named "LabSocket Server v3.5.ova" from the link on the LabSocket Download page: http://labsocket.com/download.html#LabSocketServerSoftware.

   The LabSocket Server VM OVA file contains a Virtual Machine with Ubuntu 13.10 operating system, Apache HTTP server, ActiveMQ message broker and administrative software. All of this software is preconfigured and will automatically begin execution when the VM is started.

3. Start VirtualBox

4. Select **File > Import Appliance...**

5. Navigate to the location of the OVA file and import it

6. From the Oracle VM Virtual Manager window, select the newly imported server and press the **Start** button. A new terminal window will open.

7. At the prompt, log in the VM using the username “labsocket” and password “labsocket” (no quotes for either value).

8. To enhance the security of the VM, change your password by entering the command

   ```
   passwd
   ```

   Enter and re-enter a new password of your choice when prompted.

9. Type the following command to obtain the IP address of the VM

   ```
   ifconfig
   ```

   Note the IP address for the "eth0" interface. This is the IP address that will be used later for communication with the LabSocket software on this VM.

10. The HTTP server, message broker and administrative software are all launched automatically when the VM is started. There is therefore no additional work required in the VM. End this terminal session using the command

    ```
    logout
    ```

11. The LabSocket Server VM is now ready for use and will continue to operate in the background until it is stopped using the VirtualBox Manager window.

Note – The LabSocket Server Connection Test Utility (Section 4.4) may be used to troubleshoot issues with installing and configuring the LabSocket Server.

4.2 LabSocket Software Installation

1. Start LabVIEW on your LabVIEW Host Platform.

2. Install the LabSocket VIPM package
    a. Start VIPM by selecting **Tools > VI Package Manager...**
    b. In the JKI Package Manager window select the VIPM file named **LabSocket**
    c. In the LabSocket Package window, click **Install**
    d. VIPM will display **LabSocket** and the **OpenG LabVIEW ZIP Library** and its dependencies in a list of Products to be installed. Click **Continue**
    e. After installation, **LabSocket** and the **OpenG LabVIEW ZIP Library** and its dependencies will be displayed in a list of results of the last action. Click **Finish**
    f. In the LabSocket Package window, select **File > Close**
    g. The LabSocket package should now be visible in the JKI VI Package Manager window.
    h. In the JKI VI Package Manager window, select **File > Exit**
    i. Exit LabVIEW

3. Restart LabVIEW. Installation of the LabSocket package is now complete.

Alternate VIPM File Source

NOTE - The LabSocket VIPM file can optionally be downloaded from the LabVIEW Tools Network FTP directory:
http://www.ni.com/gate/gb/GB_EVALTLKTLABSOCKET/US

In this case, substitute Step 2b above with the following two steps:

In the JKI Package Manager window select **File > Open Package File(s)**.
Select downloaded vip file and click **Open.**

4.3 Uninstalling LabSocket Software

1. Start LabVIEW

2. Remove LabSocket VIPM Package
    a. Start VIPM by selecting **Tools > VI Package Manager...**
    b. In the LabSocket Package Manager window, scroll through the list of packages to find the **LabSocket** package
    c. Right-click on the **LabSocket** package
    d. Select **Uninstall…**
    e. VIPM will display LabSocket as the only item in a list of Products to be un-installed. Click **Continue**.
    f. After installation, LabSocket will be shown as having been un-installed. Click **Finish**
    g. In the JKI VI Package Manager window, select **File > Exit**
    h. Exit LabVIEW

3. Restart LabVIEW. Uninstallation of the LabSocket evaluation package is now complete.

Note that the **OpenG LabVIEW ZIP** library and its dependencies may optionally be uninstalled as part of the uninstall process.

4.4 LabSocket Server Connection Test Utility

The LabSocket Server Connection Test utility can be helpful for troubleshooting connection issues between LabVIEW and the LabSocket Server Virtual Machine which may occur during initial system setup.

The utility is available after LabSocket installation (Section 4.2) by selecting **Tools > Bergmans Mechatronics > LabSocket > LabSocket Server Connection Test...** . Instructions for use of this utility are included within the application (Figure 4.1).

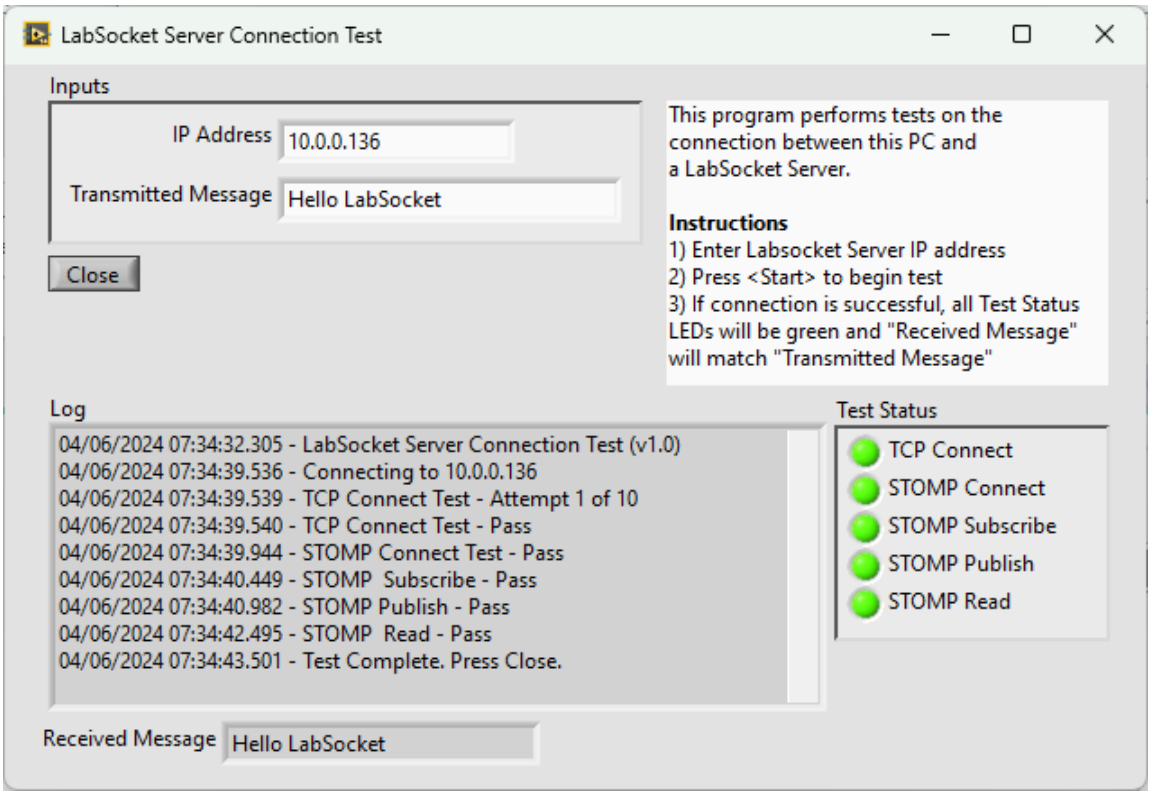


Figure 4.1. LabSocket Server Connection Test Utility

5. LabSocket Application Development

This section describes the process of developing desktop applications that operate within the LabVIEW development environment using either Basic and MultiClient client mapping modes. For an overview of client mapping refer to Section 3.2, *Client Mapping Modes*.

5.1 Creating a Basic Mapping Mode Application

This section demonstrates how to use the LabSocket system to access a simple Target VI in Basic Mapping mode from a browser.

1. Create a new LabVIEW VI. From the main LabVIEW window, select **File > New VI**

2. A key component of the LabSocket system is a VI named ***LabSocket Start.vi***. This VI calls the LabSocket Support VIs which in turn
   i) perform the front panel screenscrape; and,
   ii) synchronize the front panel and browser controls and indicators.
   Right-click in the block diagram of the new VI and select
   Bergmans Mechatronics > LabSocket > LabSocket Start.vi (Figure 5.1).
   Then, drag and drop this VI into the block diagram of the new VI.

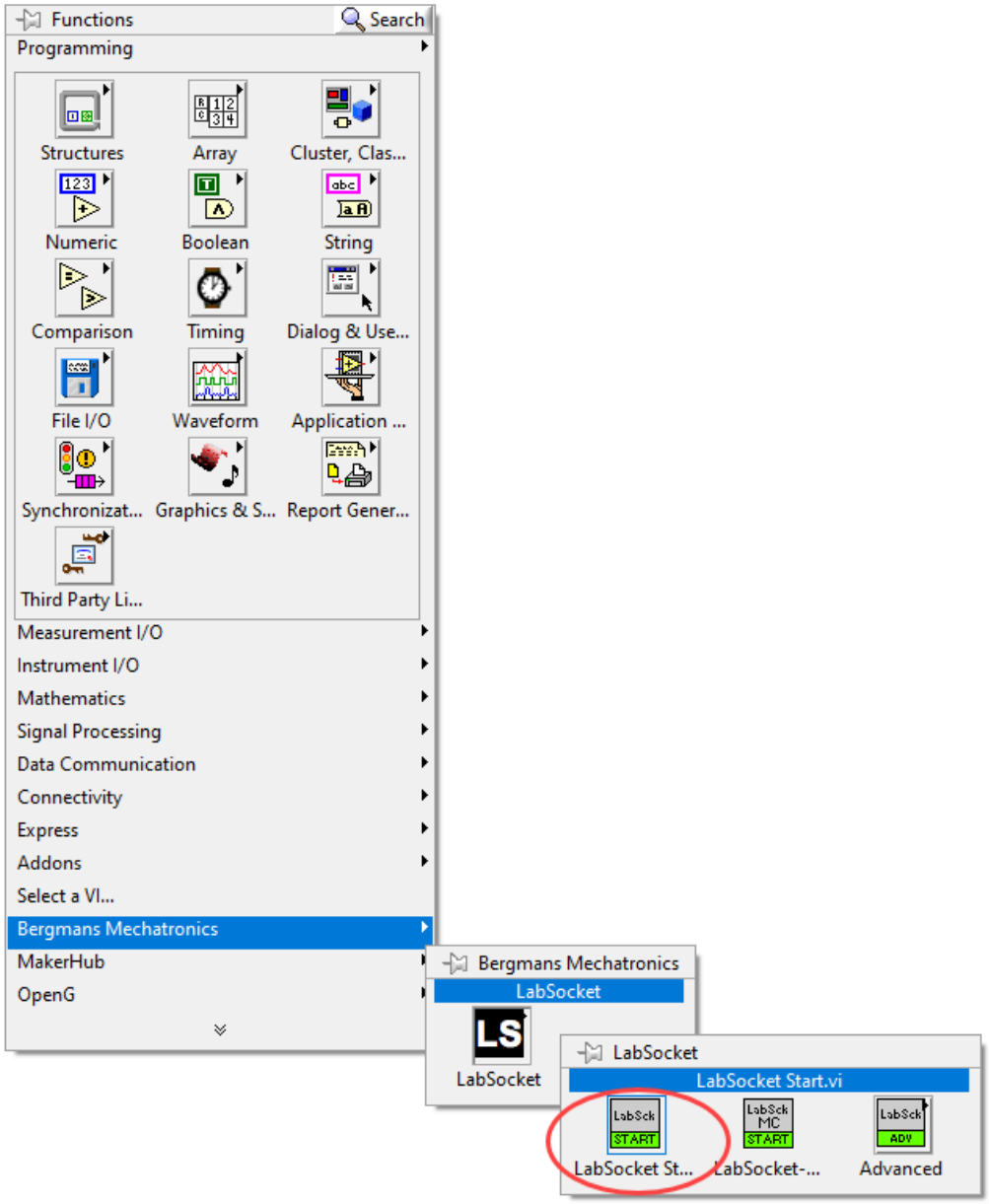


Figure 5.1. Selecting LabSocket Start.vi *from Functions Palette*

3. Wire the following three terminals of ***LabSocket Start.vi*** (Figure 5.2):
   - **IP Address** – a cluster containing two strings is wired to this input:
     - *LabSocket_Server* is the IP address of the LabSocket Server. Two options are available here:

       i) Enter the IP Address of the LabSocket Server Virtual Machine relative to the LabVIEW Host platform. (Recall that this IP address value was obtained in Section 4.1, step 9).

       OR

       ii) Enter the string "labsocket.com" (without quotes) to use the LabSocket Server software on Bergmans Mechatronics' existing cloud server. This option is intended for users who want to quickly start using the system and not for production purposes.

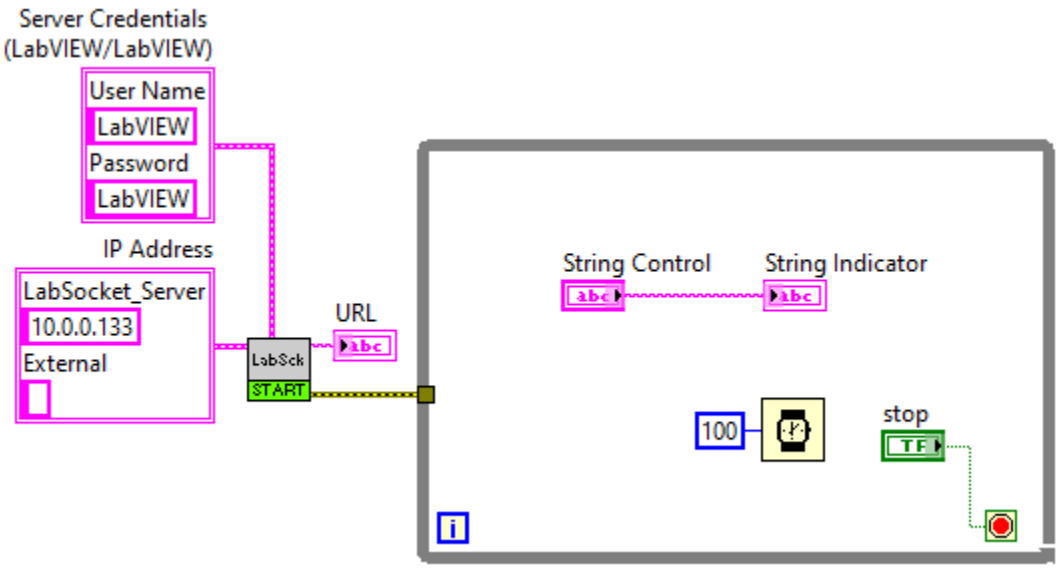
     - *External* is the IP address of the LabSocket Server VM relative to the browser. Leave this value blank if this address is the same as the *LabSocket_Server* IP address or you're using Bergmans Mechatronics' cloud server.

   - **Server Credentials** – a cluster of two text strings each containing the string "LabVIEW" is wired to this input. This is the default value for this input and matches the default settings of the message broker. (See Section 11, *User Administration,* for instructions on changing the message broker credentials)

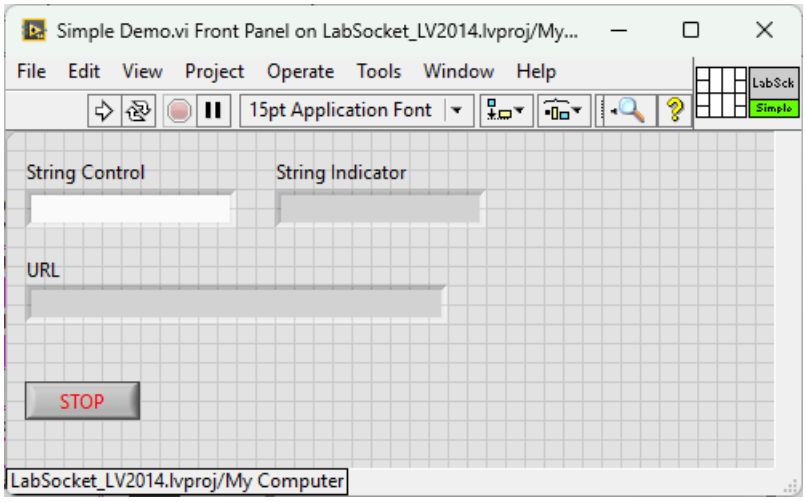
   - **URL** – wire a string indicator to this terminal. This indicator will display the URL at which the Target VI may be accessed with a browser. The URL is based on the filename of the VI. If "labsocket.com" is used as the LabSocket Server IP address, a random four-digit number is appended to the URL.

   Details about the terminals of ***LabSocket Start.vi*** are available in Section 8, *LabSocket Start VIs*.

4. Add a while loop, a Wait VI, and a string control and indicator and stop button as shown below (Figure 5.2). Wire the **error out** terminal of ***LabSocket Start.vi*** to the edge of the while loop. This wire forces ***LabSocket Start.vi*** to complete execution before the while loop starts.



a) Block Diagram



b) Front Panel

Figure 5.2. Simple Demo.vi

5. Set the mechanical action of buttons to either "Switched when Pressed" or "Switched when Released".

6. Save the file to disk. In this guide, the filename ***Simple Demo.vi*** is used, although any name may be used.

7. Start the VI by pressing the VI **Run** button.

8. The **LabSocket Status** window will appear. The URL at which the Target VI may be accessed will be displayed in the Log page of this window (Figure 5.3). This

URL will match that output on the URL terminal of ***LabSocket Start.vi***. See Section 9, *LabSocket Status Window*, for additional information about the LabSocket Status window

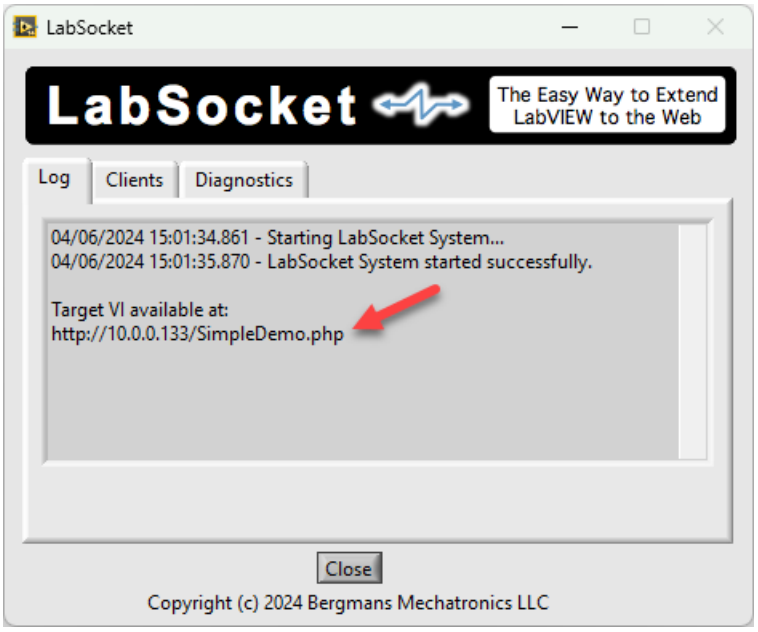


Figure 5.3 LabSocket Status Window (Log Page)

9. When a browser is pointed to the URL displayed in the **LabSocket Status** window, it will automatically display a representation of the Front Panel. All supported controls and indicators in the front panel will also become synchronized with those in the browser (Figures 5.4 and 5.5).

   Each browser that connects to the system via the URL of the web client will contain the same content as the first browser. Each user also has the ability to change any of the active controls in the browser.

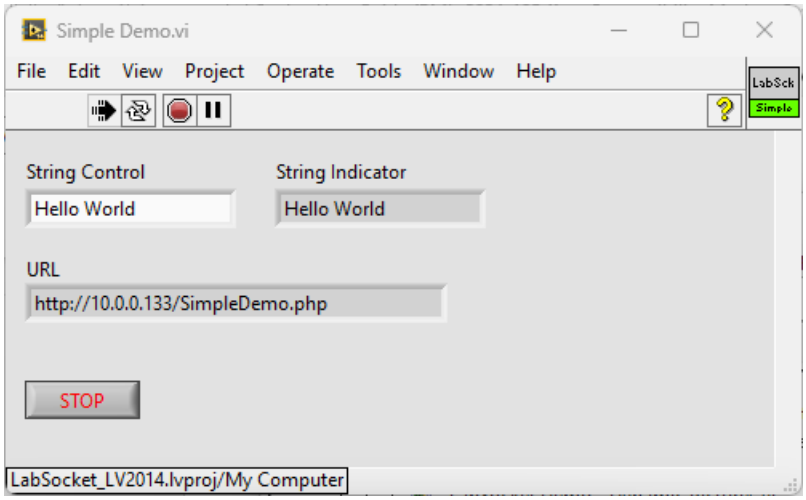


Figure 5.4. Front Panel of Target VI "Simple Demo.vi"

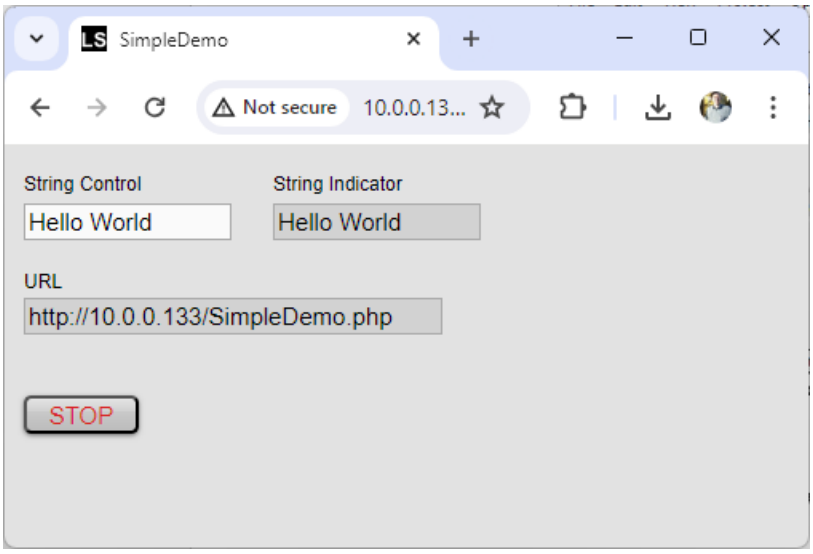


Figure 5.5. Browser Representation of Target VI "SimpleDemo.vi"

5.2 Creating a MultiClient Mapping Mode Application

5.2.1 MultiClient Mapping Mode – How it Works

In MultiClient Mapping mode, each browser client connects to a unique instance of the Target VI. With reference to Figure 5.6, this mapping mode operates as follows: A "Server" VI is the top level VI in the application. The "LabSocket-MC Start" VI on the block diagram of this VI is used to invoke MultiClient Mapping mode

"LabSocket-MC Start" VI is similar to the "LabSocket Start" VI described in the Basic Mapping mode section but includes two additional inputs: 1) the path to the Target VI; and, 2) the maximum number of browser clients that may connect to the system.

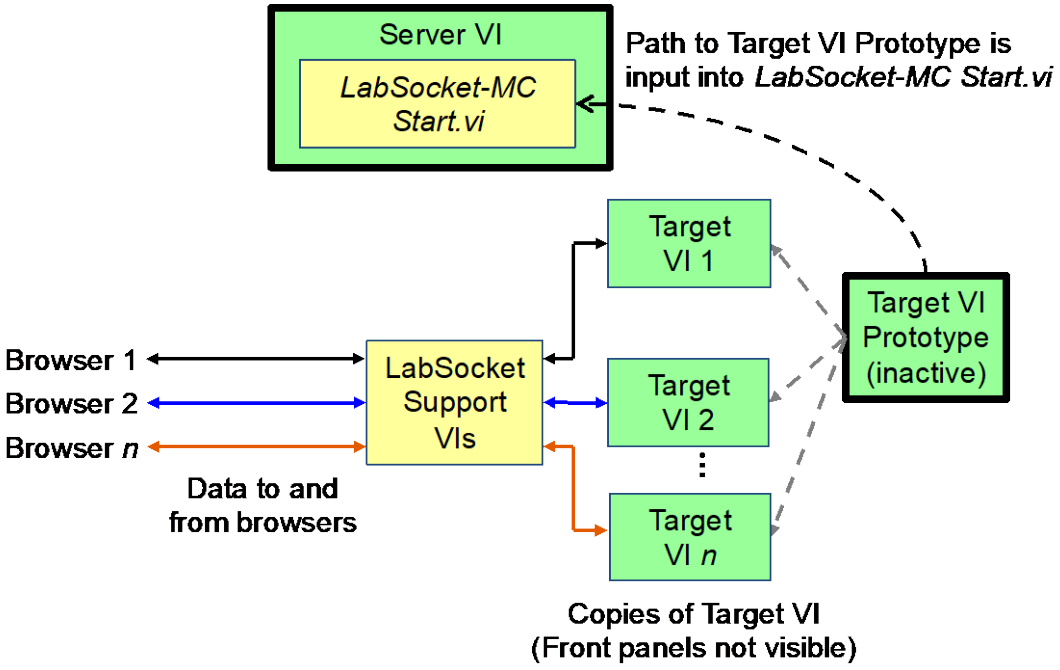


Figure 5.6 Overview of MultiClient Mapping Mode

As browsers connect to and disconnect from the system, LabSocket starts and stops unique instances of the Target VI. When a browser is connected to the system, LabSocket automatically synchronizes each browser with a unique Target VI instance. The significance of MultiClient mapping mode is that is enables each remote user to interact in an independent session with an instance of the Target VI

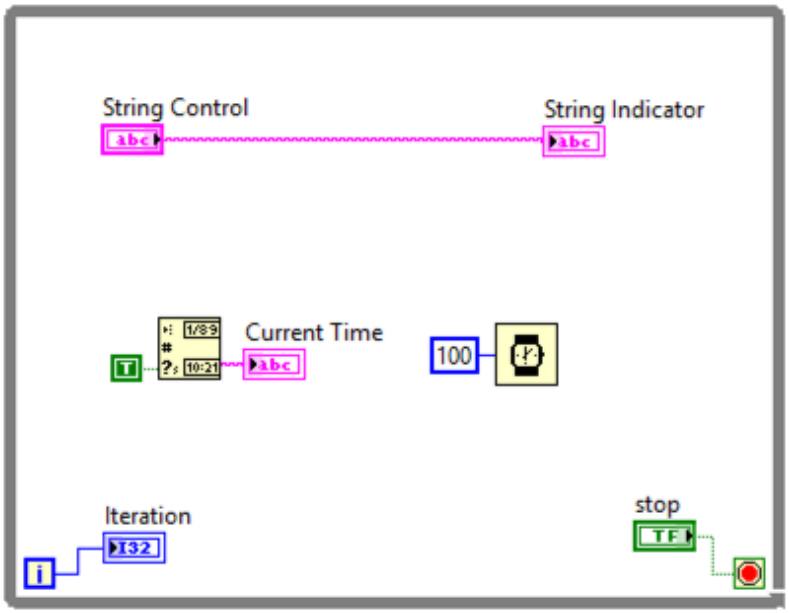
5.2.2 Step-by-Step Instructions

The following instructions show how to use LabSocket to access a Target VI in MultiClient Mapping mode. The instructions consist of creating two VIs: 1) a simple

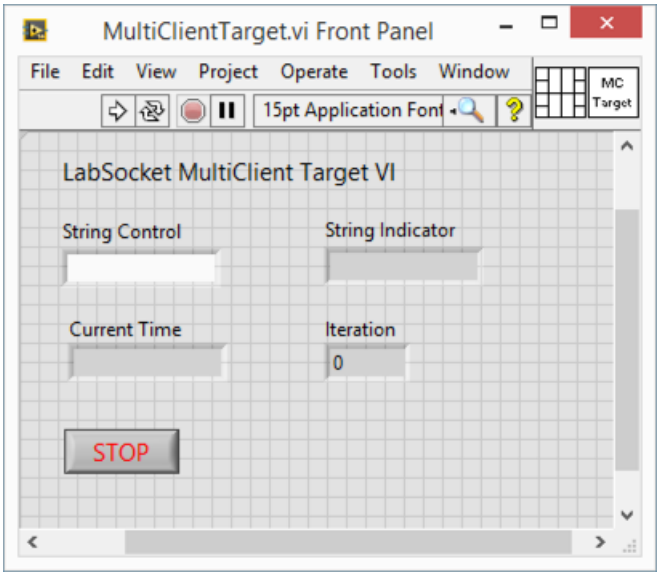
Target VI; and, 2) a top-level Server VI that calls the LabSocket software and identifies the Target VI.

1. Create a new **Target VI** that contains a while loop, a Wait VI, a Get Date/Time String VI, and controls and indicators as shown below (Figure 5.7). It is important to note that in MultiClient Mapping mode, the Target VI does not contain any LabSocket VIs.

2. Set the mechanical action of the Stop button to either "Switched when Pressed" or "Switched when Released". Save the file to disk. In this guide, the filename ***MultiClientTarget.vi*** is used, although any name is acceptable.



a) Block Diagram



b) Front Panel

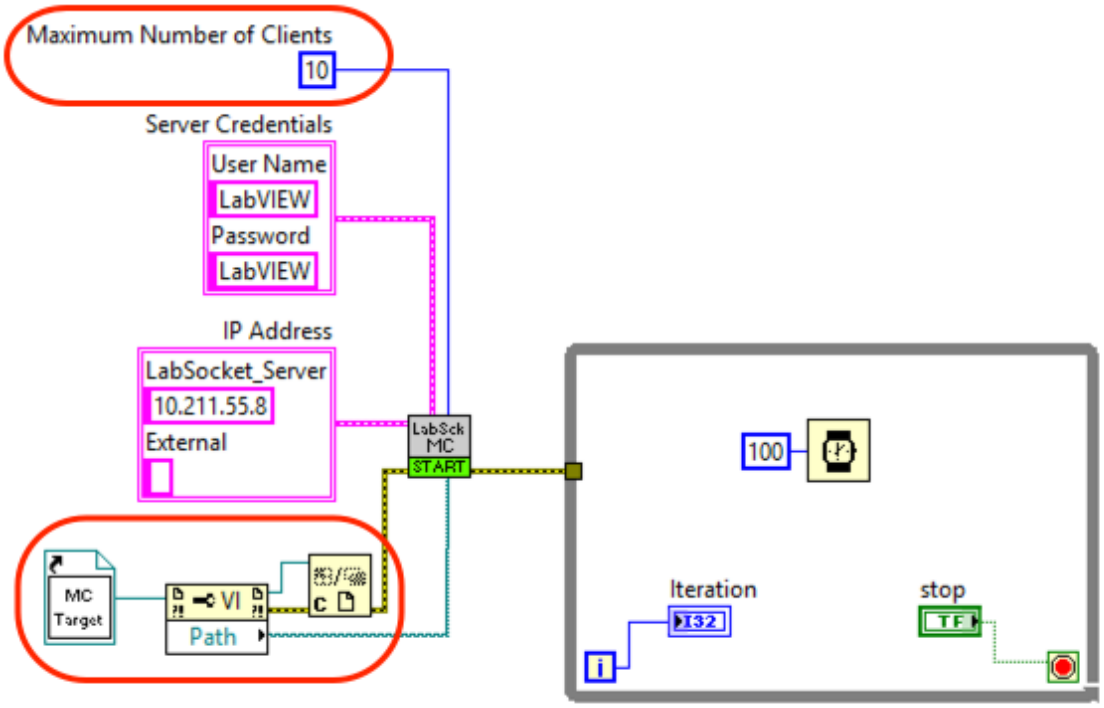
Figure 5.7. MultiClientTarget.vi

3. Create a new **Server VI**. Add to this VI a while loop, a Wait VI, and controls and indicators as shown below (Figure 5.8).

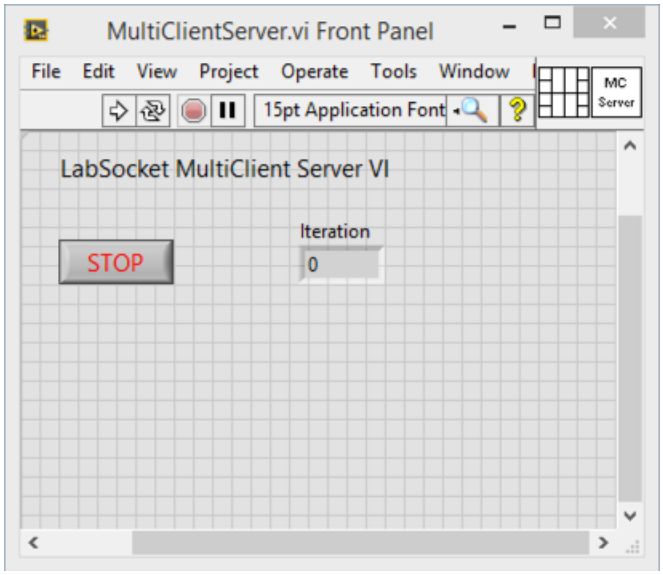
4. Also add the ***LabSocket-MC Start.vi*** to the block diagram of the Server VI. This VI can be found in the Tools Palette under
   Bergmans Mechatronics > LabSocket (Figure 5.9).

5. ***LabSocket-MC Start.vi*** has the same terminals as ***LabSocket Start.vi*** plus two additional terminals:
   - **Maximum Number of Clients** – This terminal specifies the maximum number of browsers that can connect to the system simultaneously. For this application, set this input to 10.
   - **Target VI** – This terminal specifies the path to the Target VI. For this application, add i) a reference to the Target VI created above; ii) a VI Path property node; and, iii) a Close Reference VI and wire these elements together (See the block diagram in Figure 5.8).

6. Save the **Server VI**. In this guide the filename ***Server.vi*** is used, though any name may be used.



a) Block Diagram



b) Front Panel

Figure 5.8. MultiClientServer.vi

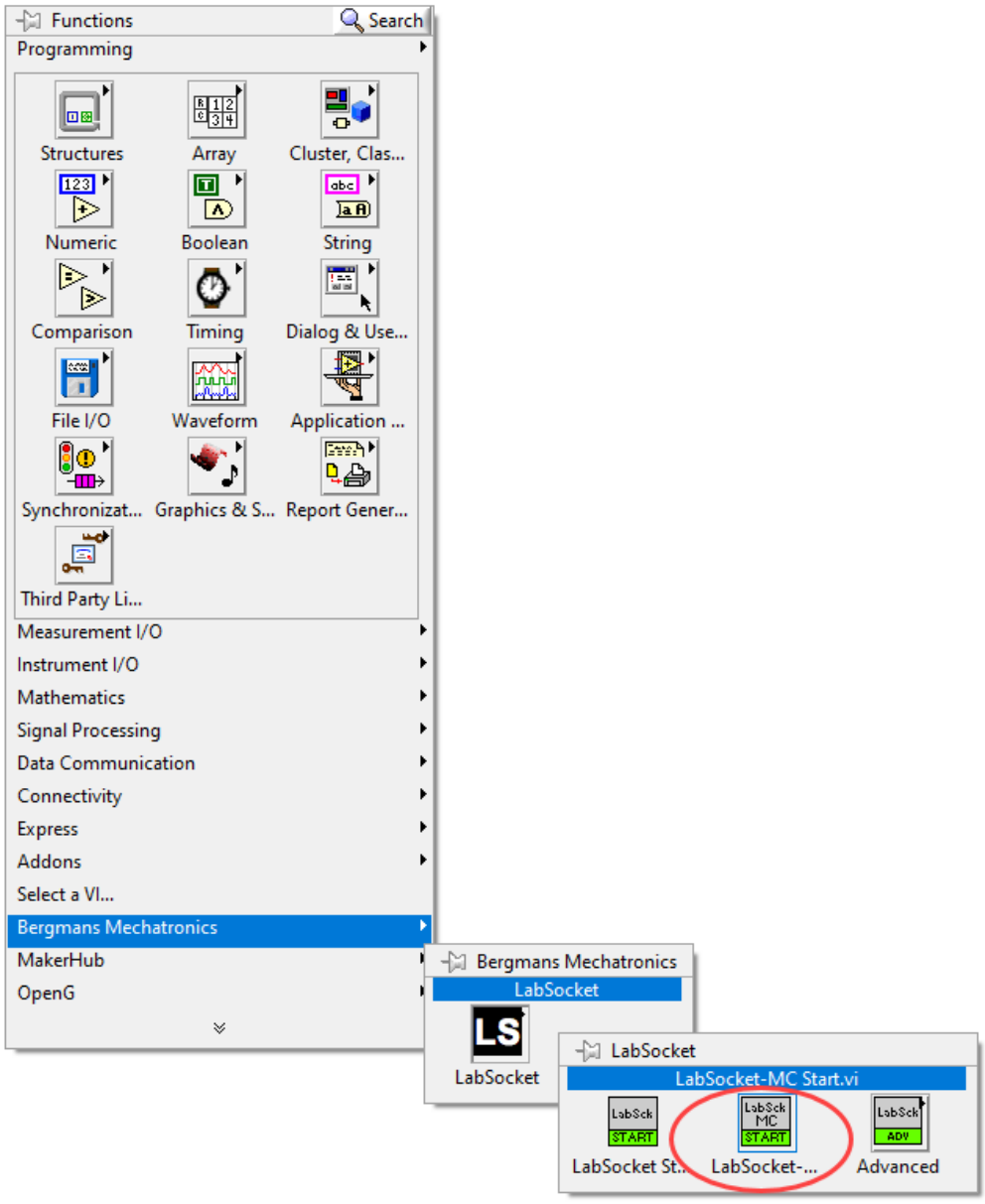


Figure 5.9. Selecting LabSocket-MC Start.vi *from Functions Palette*

7. Start the application by pressing the ***Run Arrow*** in the ***Server.vi*** front panel. Note that ***MultiClientTarget.vi*** is not running and will remain in this state for the duration of this exercise.

8. Point several browsers to the URL displayed in the ***LabSocket Status*** window. Each browser will be connected to a unique instance of the Target VI as shown in Figure 5.10. These instances are copies of the Target VI that are created automatically as peers to the Target VI for each browser client. The file name of each instance is the name of the Target VI with the characters "_XXX" appended to the name, when XXX is a unique ID number for the browser client. When a browser client disconnects from the system, its corresponding Target VI instance is automatically deleted.

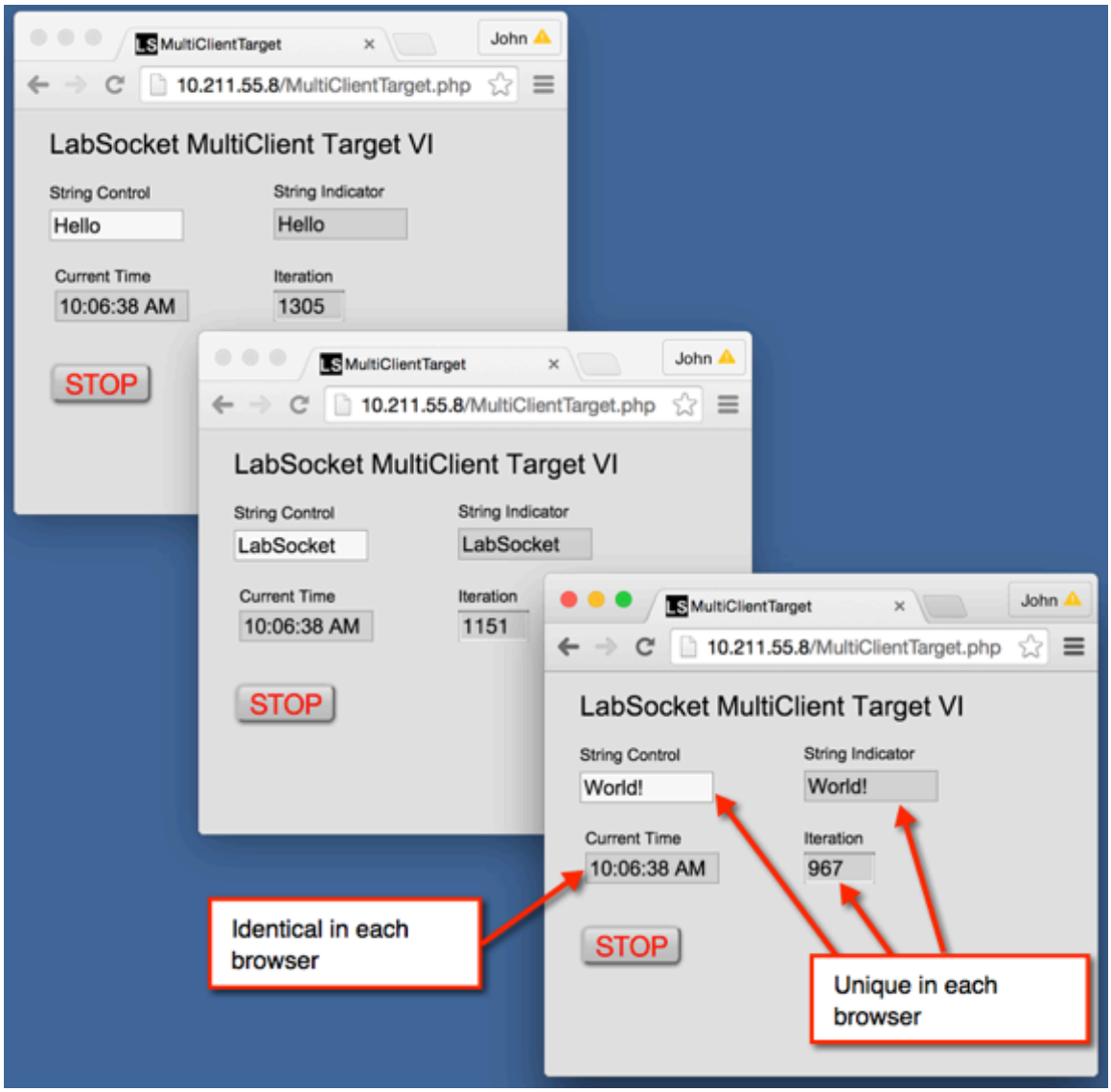


Figure 5.10. Three Browsers, Each Connected to a Unique Instance of MultiClientTarget.vi

6 Standalone Applications

Two types of standalone applications may be created using LabSocket: I) executables for Windows platforms; and, ii) deployed applications for CompactRIO 903x NI Linux Real-Time Targets. Any number of either application type may be created using a single LabSocket license.

Sections 6.1 and 6.2 describe procedure for creating standalone executables using Basic and MultiClient Client Mapping, respectively. Section 6.3 provides instructions on how to deploy a Basic Client Mapping application to a cRIO-903x platform.

6.1 Basic Mapping

This procedure demonstrates how to create a standalone Windows executable using the ***Simple Demo.vi*** Target created in Section 5.1, *Creating a Basic Mapping Mode Application*, as an example. Figure 6.1 shows the initial state of the ***Basic Mapping.lvproj*** example LabVIEW project that will be used to create the standalone executable.

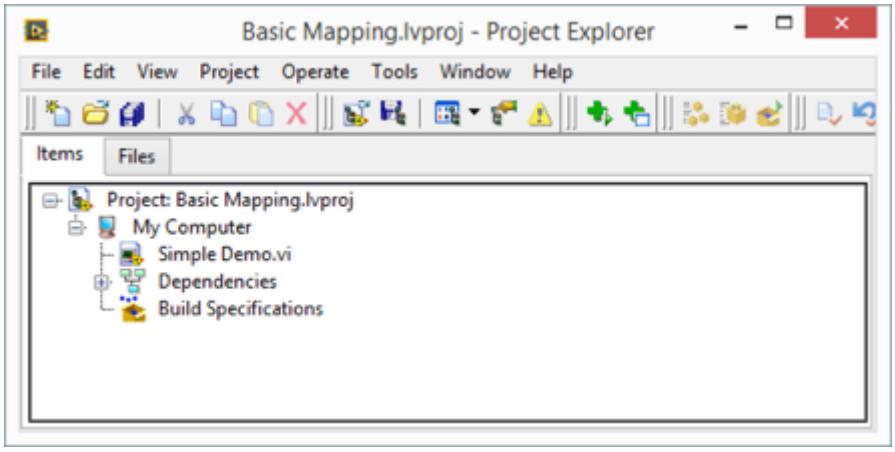


Figure 6.1. Initial State of Basic Mapping,lvproj *Project*

1. **Right-click** on Build Specifications and select **New>Application (EXE)**. This opens the Application Properties Dialog Box.

2. On the **Information** page, enter values for the **Build Specification Name, Target Filename** and **Destination Directory**. Any values may be used for these fields. The values used in this procedure are shown in Figure 6.2.

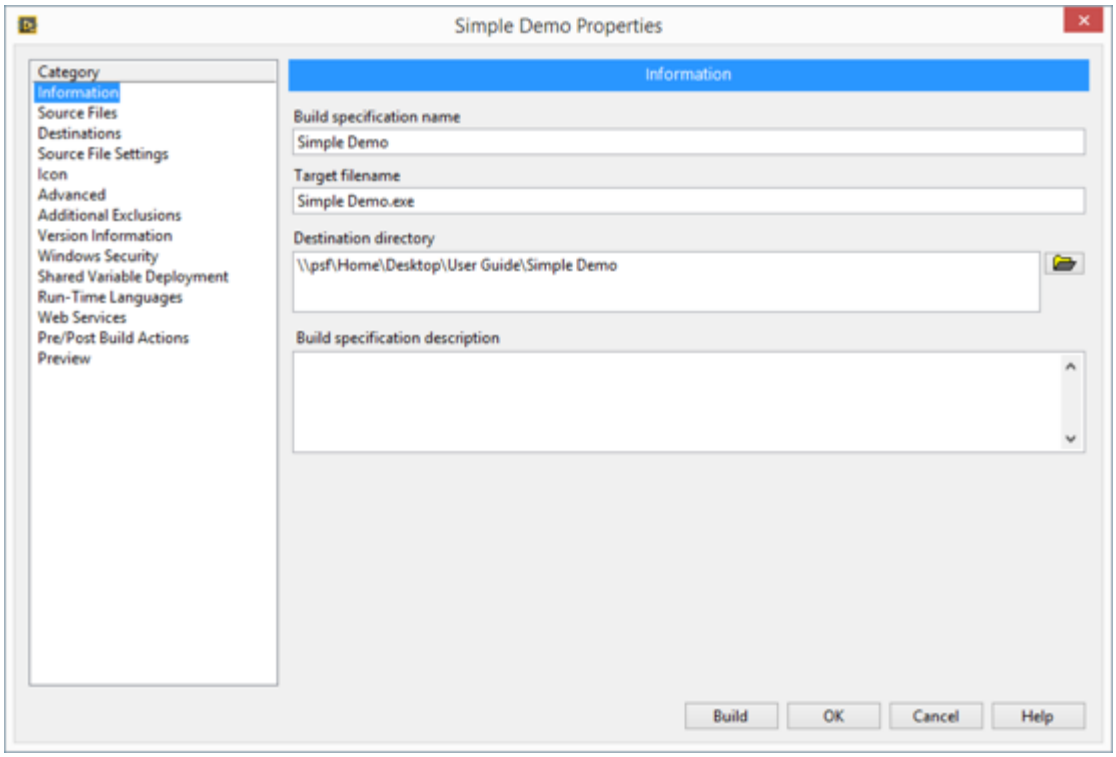


Figure 6.2. Build Specification Information Page

3. On the **Source Files** page, click on the Target VI and select it as a **Startup VI** (Figure 6.3)

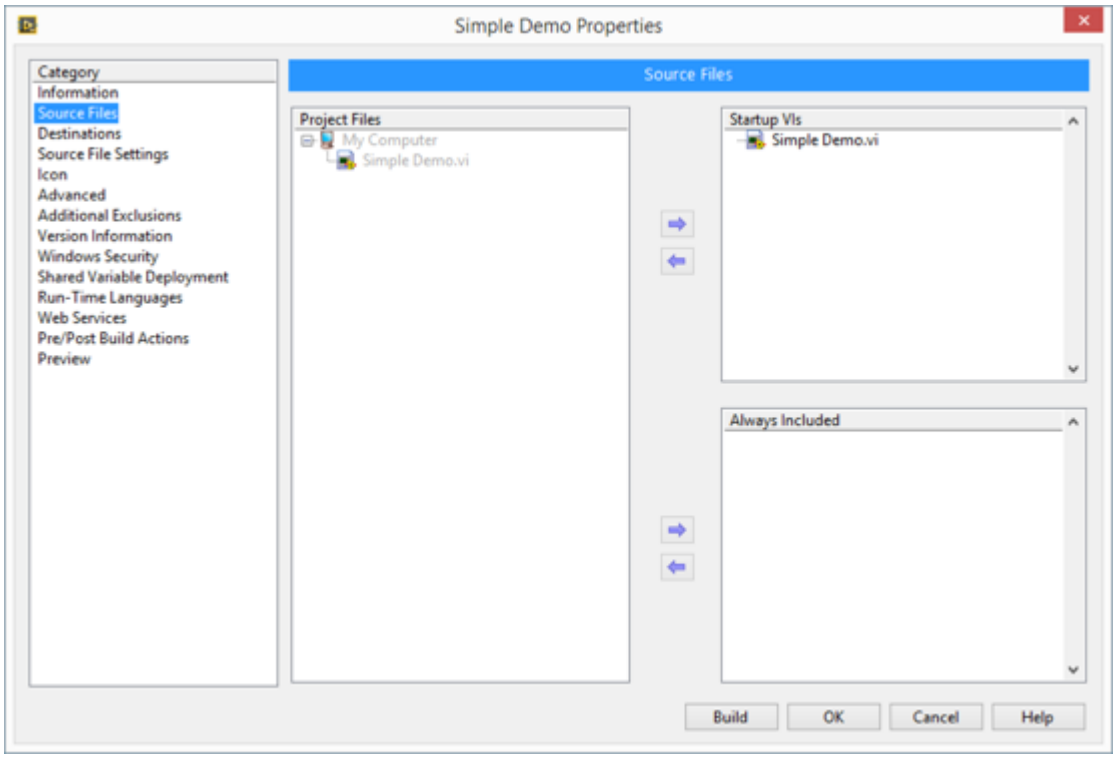


Figure 6.3. Build Specification Source Files Page

4. Review the settings on the **Destinations** page (Figure 6.4). The **Destination path** setting displayed here should match that selected on the **Information** page.

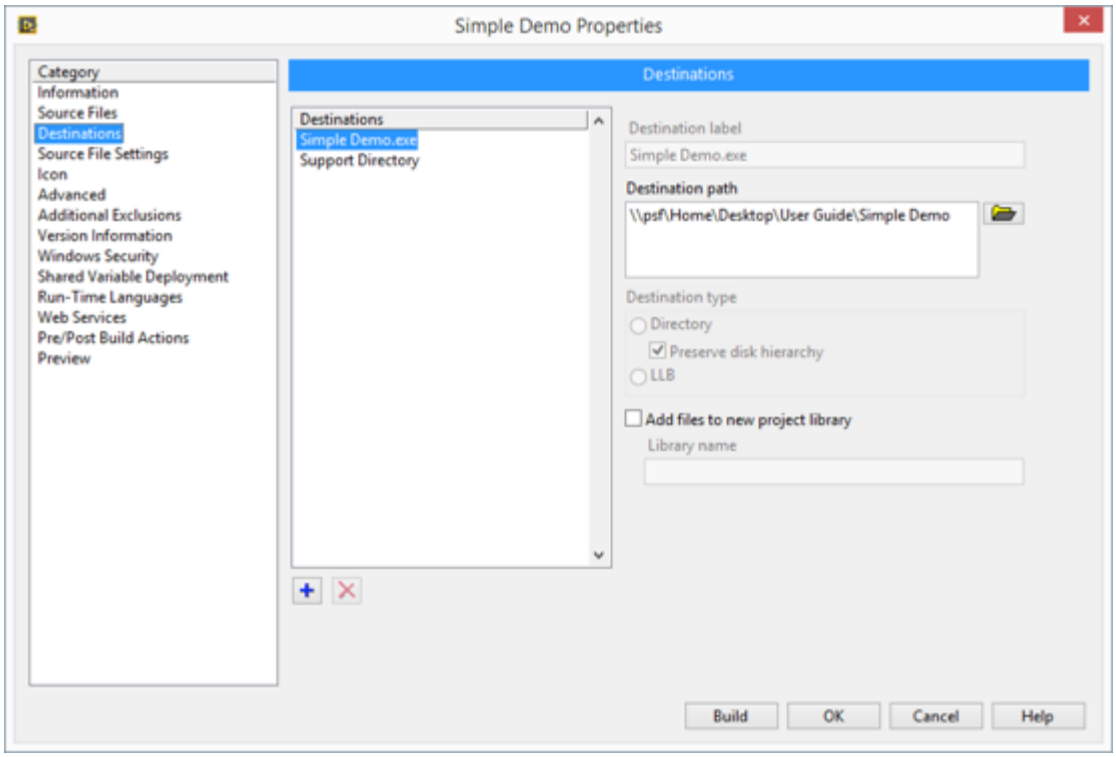


Figure 6.4 Build Specifications Destinations Page

5. Review the settings on the **Source File Settings** page (Figure 6.5). No changes are necessary here.

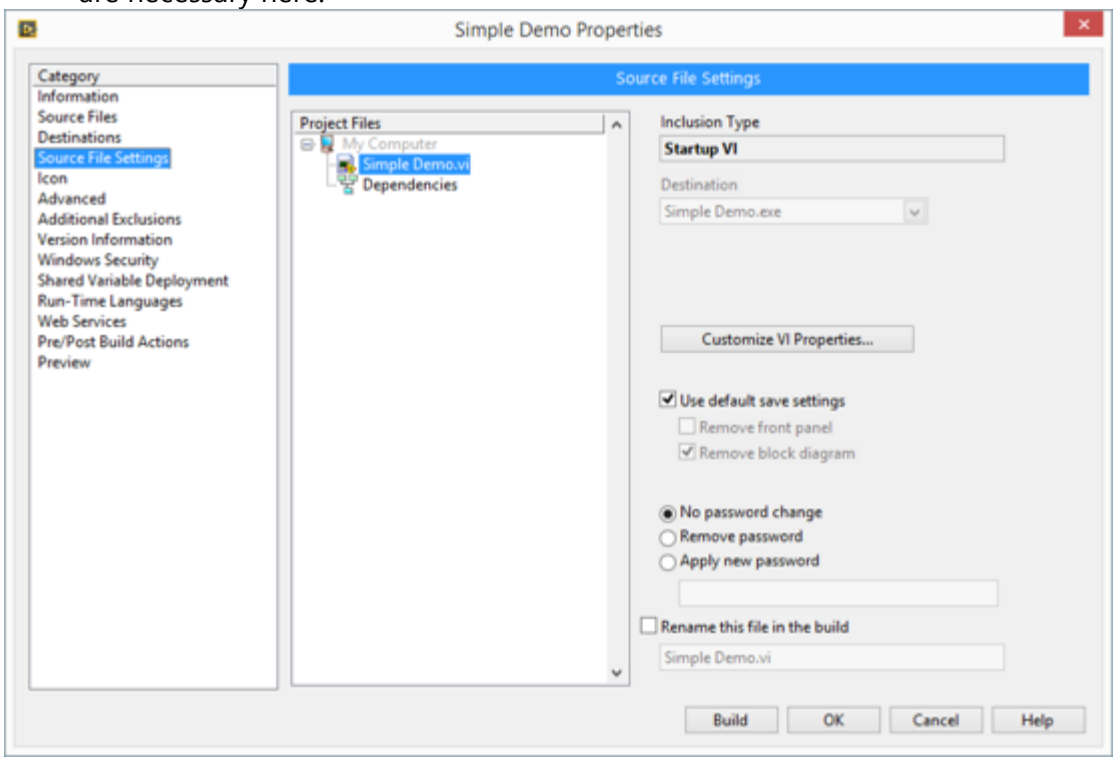


Figure 6.5 Build Specification Source File Settings Page

6. Press **Build** to build the executable. Upon completion, the executable can be found in the build destination directory (Figure 6.6). As an option, the project may be closed and LabVIEW may be exited.

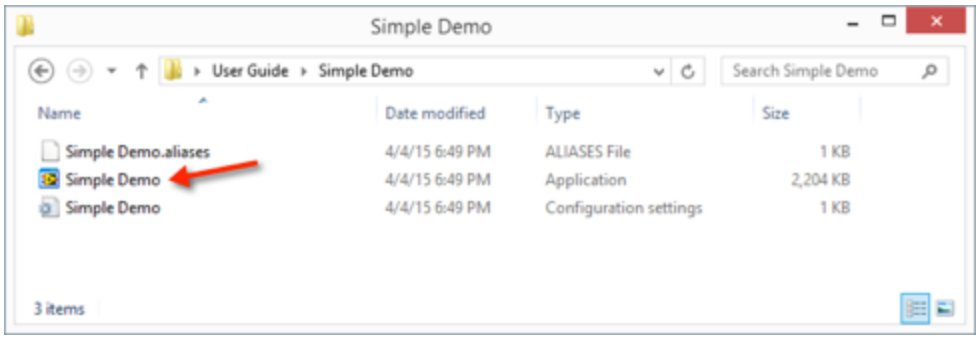
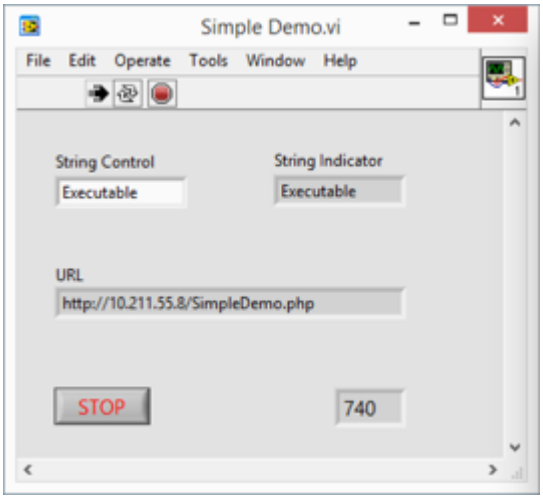
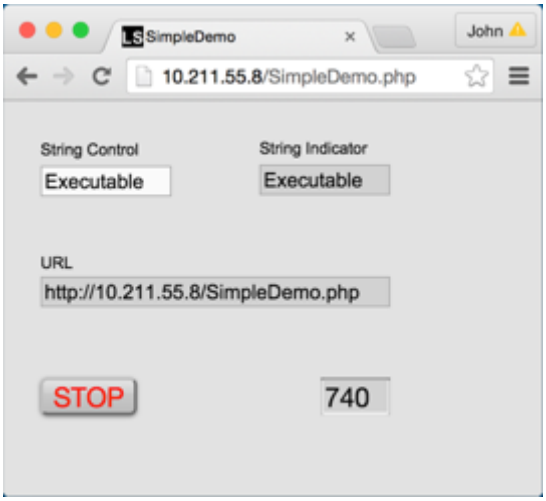


Figure 6.6. Executable Located in Build Destination Directory

7. Double-click on the application icon to run it, then point a browser to the URL in the LabSocket Status Window to access it remotely (Figure 6.7).



a) Simple Demo.vi *Operating as a Standalone Executable*



b) Accessing the Simple Demo.vi *Executable in a Browser*

Figure 6.7. Browser Access to a Standalone Basic Mapping Executable

6.2 MultiClient Mapping

This procedure demonstrates how to create a standalone Windows executable using the ***MultiClientTarget.vi*** and ***MultiClientServer.vi*** VIs created in Section 5.2, *Creating a MultiClient Mapping Mode Application*, as an example. Figure 6.8 shows the initial state of the ***MultiClient Mapping.lvproj*** example LabVIEW project that will be used to create the standalone executable.

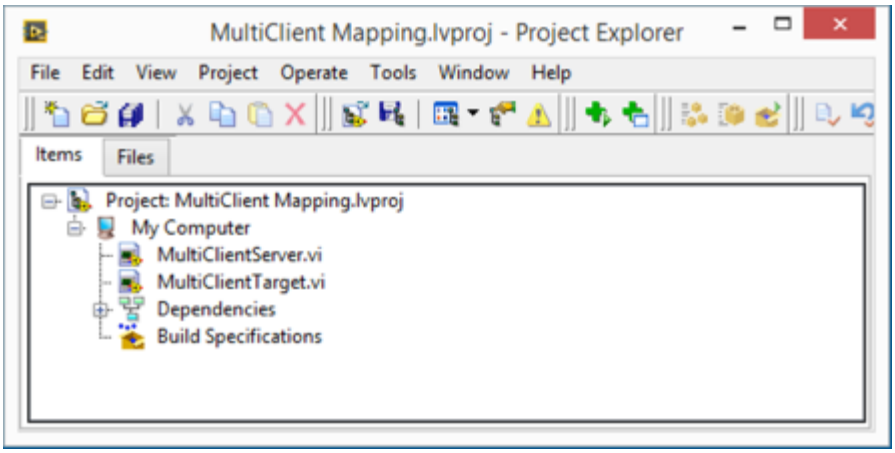


Figure 6.8. Initial State of MultiClient Mapping,lvproj *Project*

1. **Right-click** on Build Specifications and select **New>Application (EXE)**. This opens the Application Properties Dialog Box.

2. On the **Information** page, enter values for the **Build Specification Name, Target Filename** and **Destination Directory**. Any values may be used for these fields. The values used in this procedure are shown in Figure 6.9.

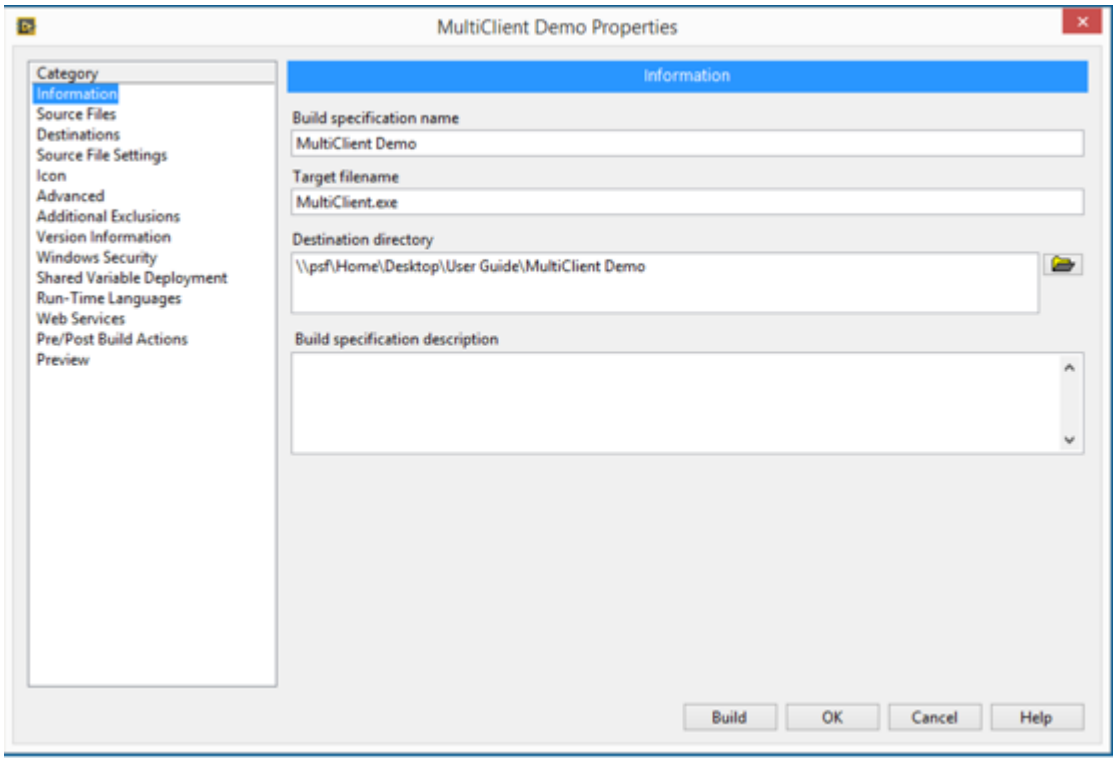


Figure 6.9. Build Specification Information Page

3. On the **Source Files** page, set **MultiClientServer.vi** as a **Startup VI** and **MultiClientTarget.vi** an **Always Included VI.** (Figure 6.10).

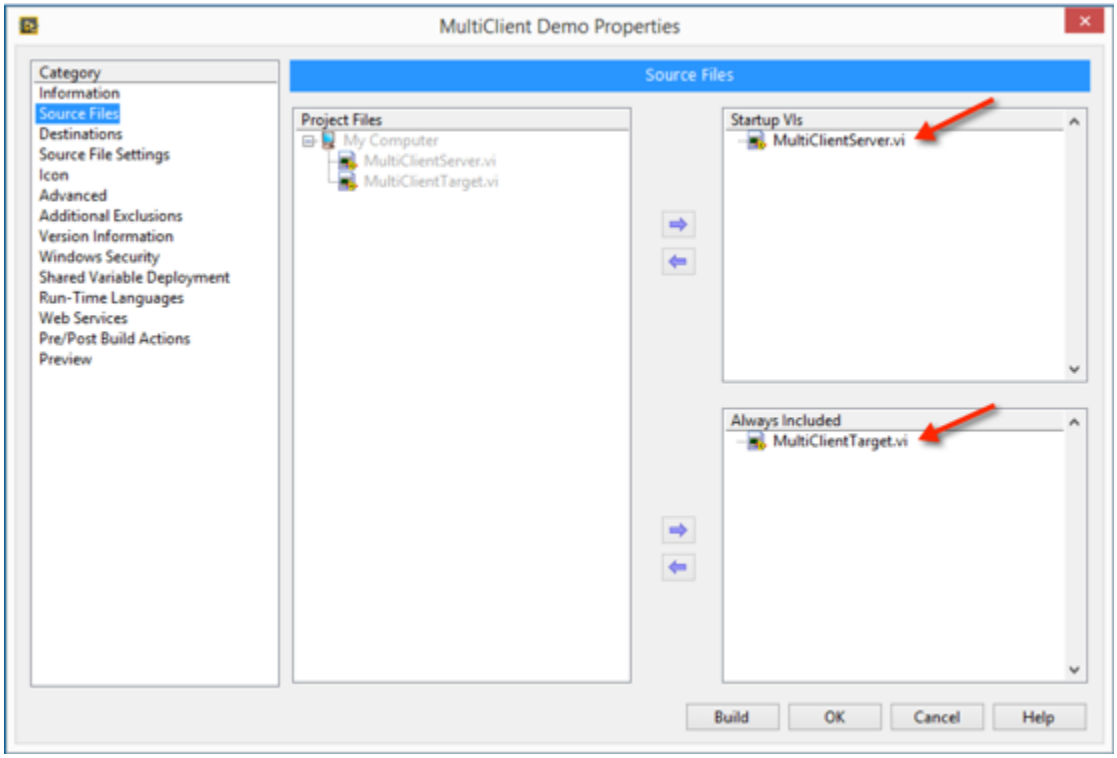
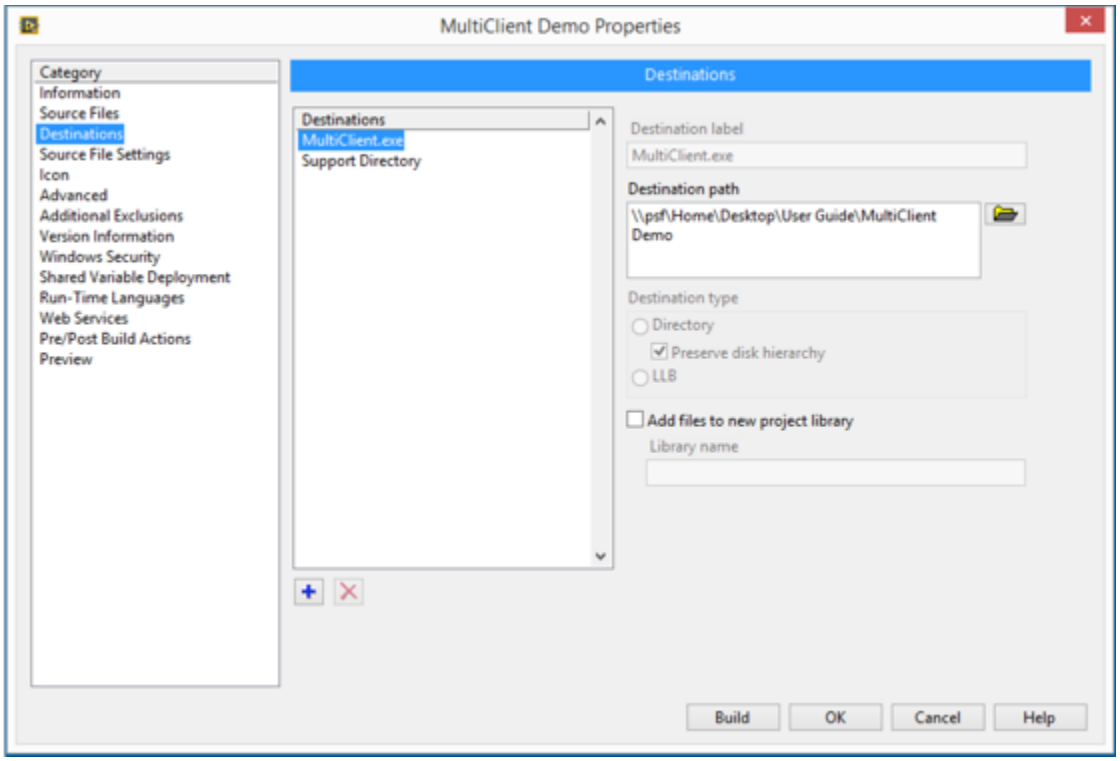
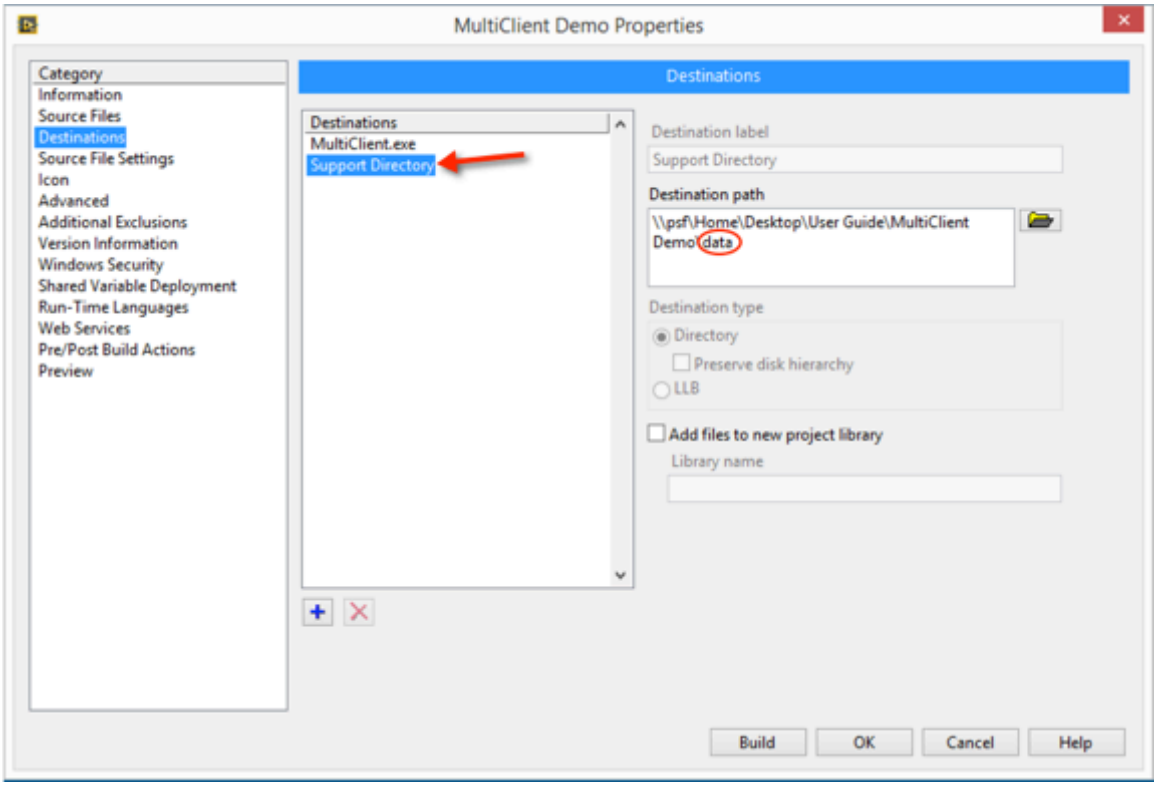


Figure 6.10. Build Specification Source Files Page

4. Select the **Destinations** Category
    a. The **Destination path** setting for the executable should match that selected on the **Information** page (Figure 6.11a).
    b. The **Destination path** setting for the "Support Directory" should be the "data" directory within the Destination directory of the executable (Figure 6.11b). This directory will be the location for the prototype Target VI and copies of this VI that are created when browsers connect to the executable.



a) Destination for MultiClient.exe



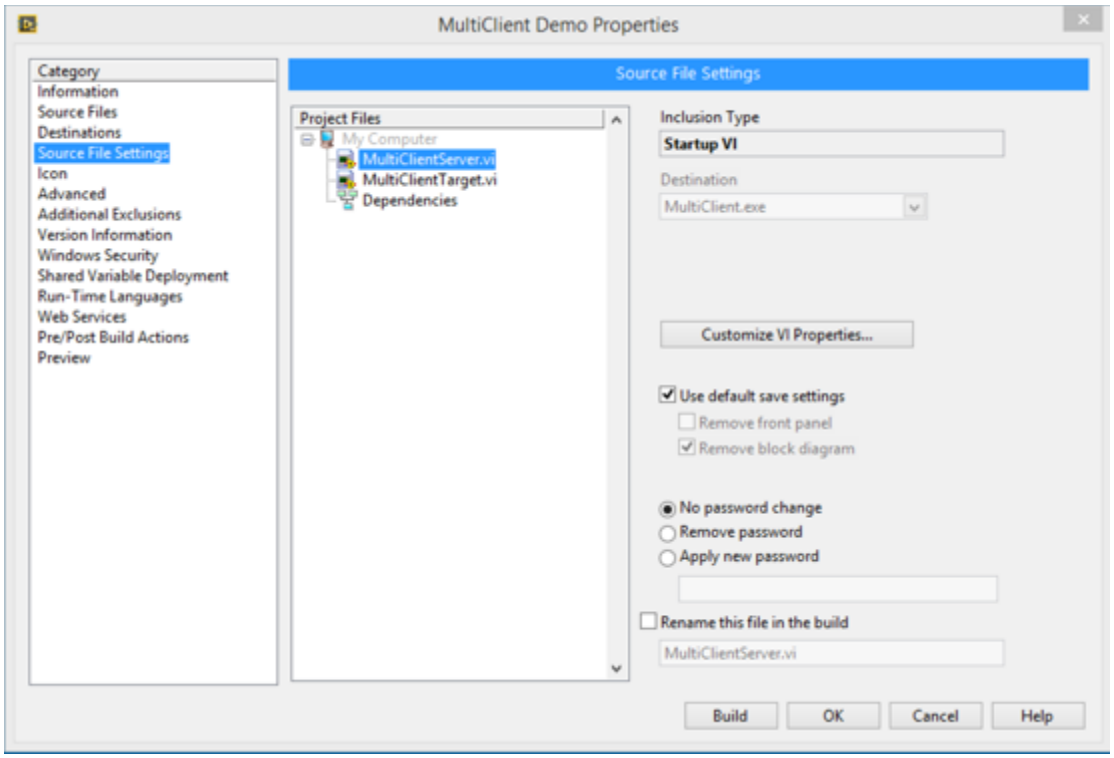
b). Destination of Support Directory

Figure 6.11. Build Specification Destinations Page

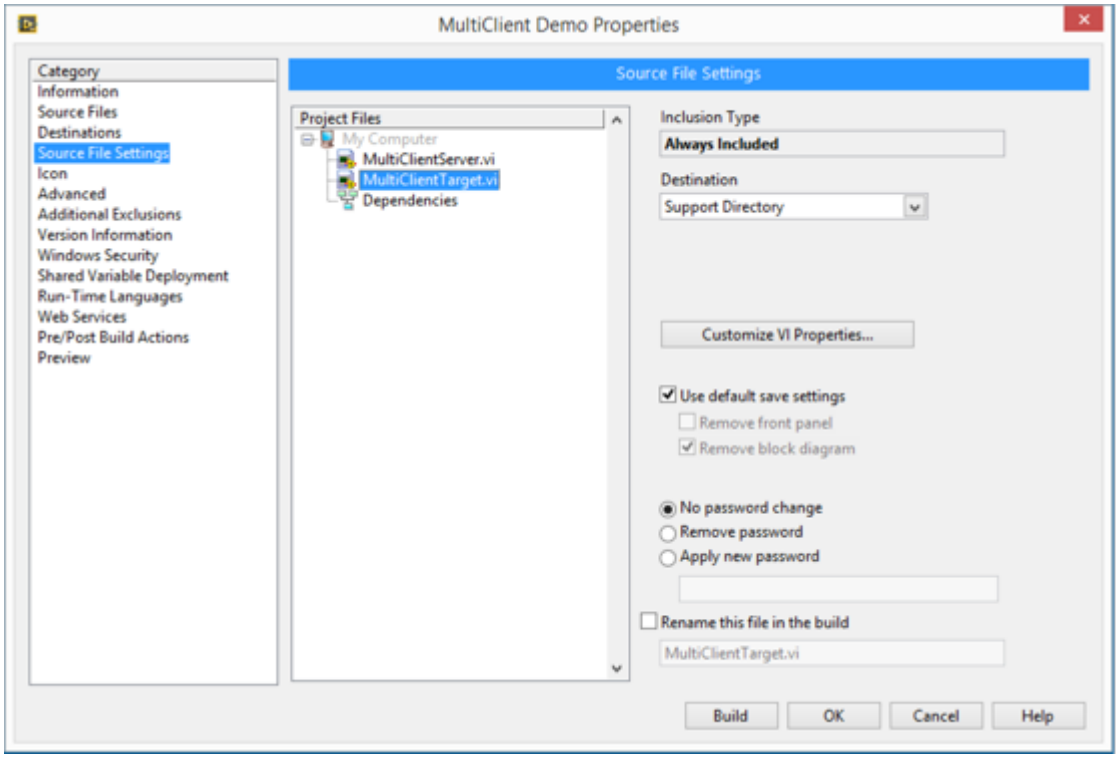
5. Review the settings on the **Source File Settings** page (Figure 6.12). The default values for "Inclusion Type" and "Destination", shown in Table 6.1, should be used.

Table 6.1. Recommended Source File Settings for Standalone MultiClient Applications

Project File	Inclusion Type	Destination
MultiClientServer.vi	Startup VI	MultiClient.exe
MultiClientTarget.vi	Always Included	Support Directory



a) Source File Settings for MultiClientServer.vi



b) Source File Settings for MutliClientTarget.vi

Figure 6.12. Build Specification Source File Settings

6. Press **Build** to build the executable. Upon completion, the executable can be found in the build destination directory (Figure 6.13). As an option, the project may be closed and LabVIEW may be exited.

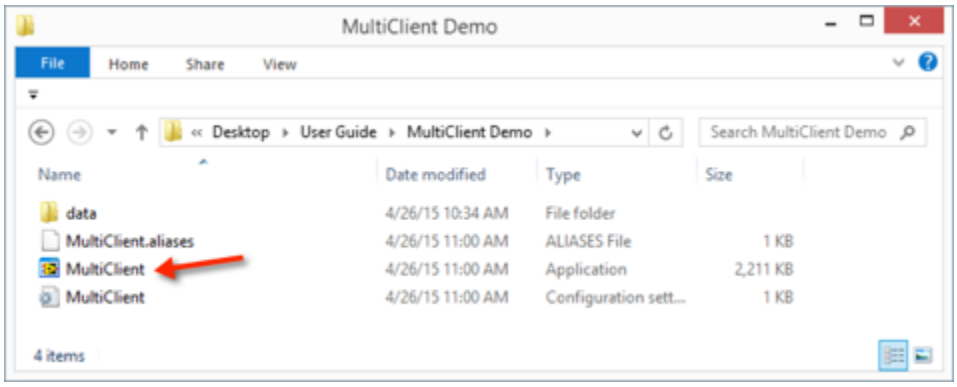


Figure 6.13. MultiClient Executable Located in Build Destination Directory

7. Double-click on the application icon to run it, then point multiple browsers to the URL in the LabSocket Status Window to establish remote connections to independent instances of the Target VI (Figure 6.14).

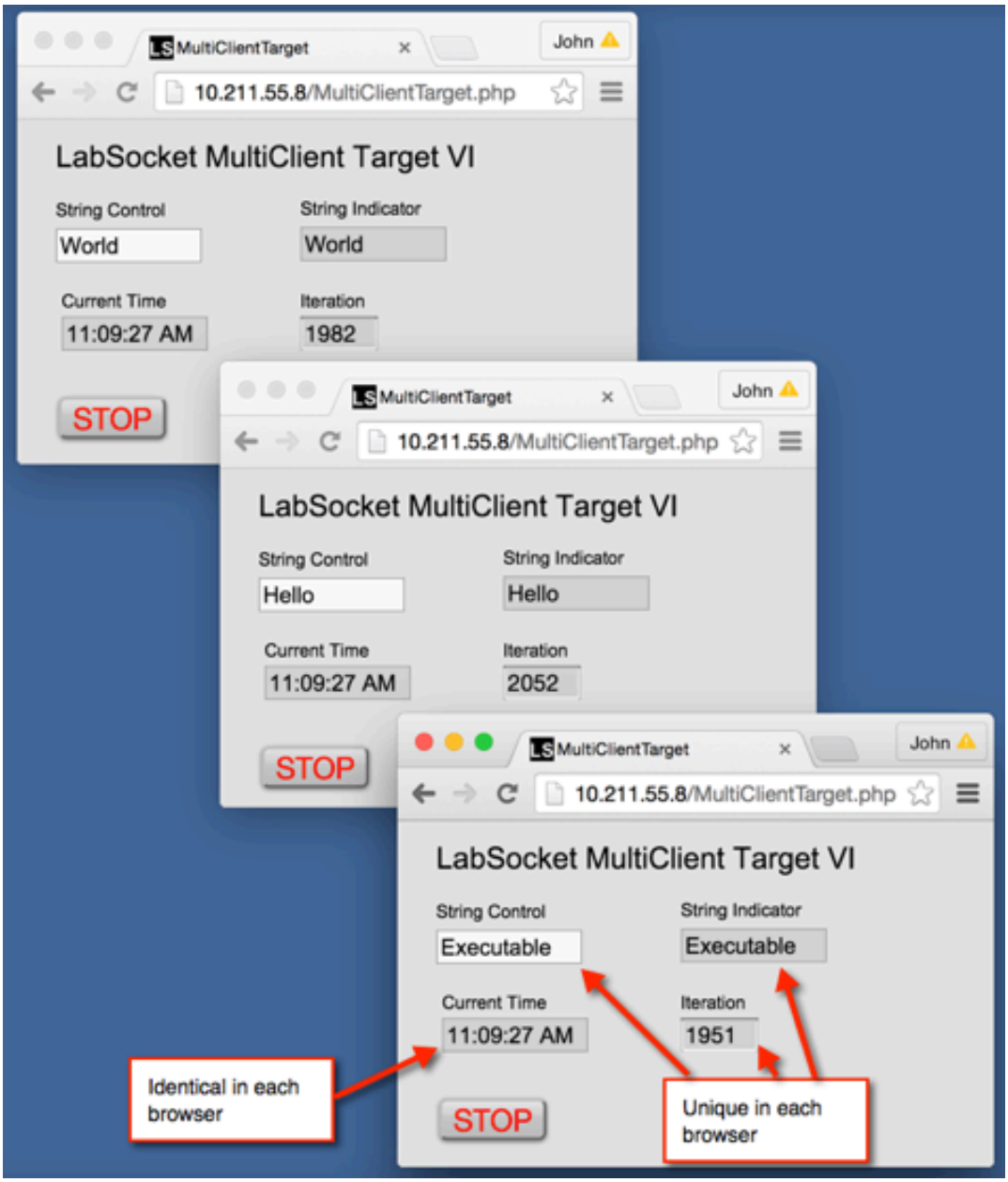


Figure 6.14. Three Browsers, Each Connected to a Unique Instance of MultiClientTarget.vi *Launched from Executable of* MultiClientServer.vi

6.3 Deploying to NI Linux Real-Time Targets

The following instructions describe how to deploy a Target VI that includes the LabSocket software to a CompactRIO cRIO-903x for operation using the LabSocket Basic client mapping mode.

Two key requirements for use of LabSocket on cRIO-903x platforms are:

- the embedded UI of the cRIO-903x must be turned on; and,
- the cRIO-903x may only operate in headless mode.

These instructions assume that the cRIO-903x device has already been connected to the development platform and that a LabVIEW project that includes the cRIO-903x device has already been created.

1. Add the Target VI to cRIO-903x target device in the LabVIEW project. In this procedure the demo VI "LabSocket Demo – Element Test.vi" will be used as the Target VI. (Figure 6.15)

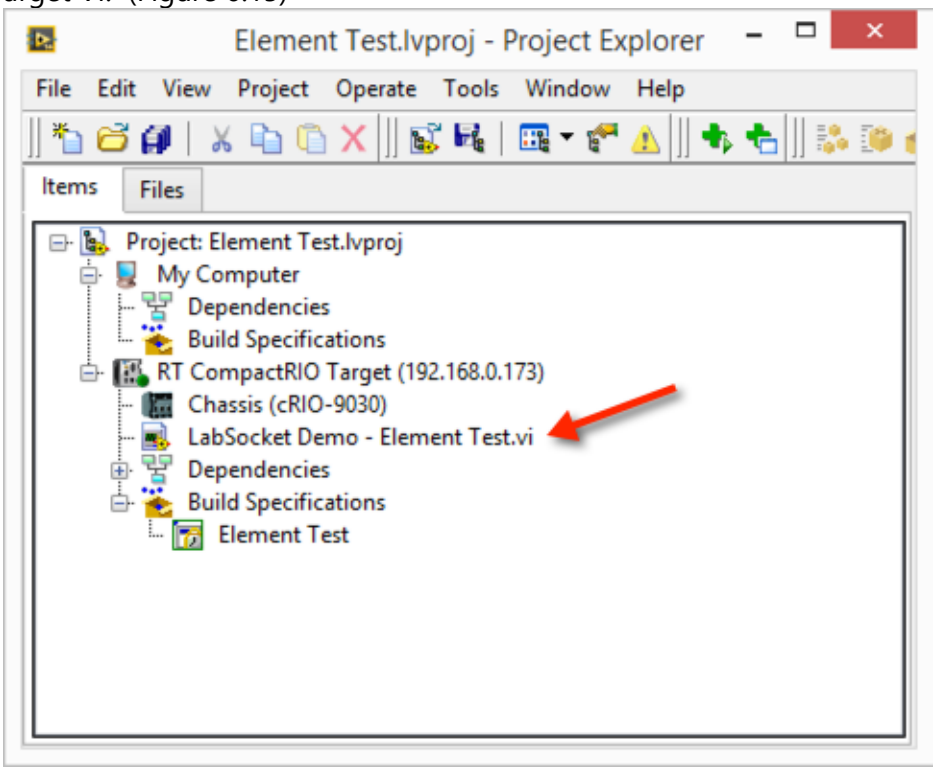


Figure 6.15. Adding Target VI LabSocket Demo – Element Test.vi *to CompactRIO cRIO-9030 Target Node in LabVIEW Project*

2. Right-click on **Build Specification** and select **New > Real-Time Application**. The Application Properties pop-up window will open.

3. Enter the **Build specification name** on the **Information** page. In this case the name is "Element Test" is used (Figure 6.16).

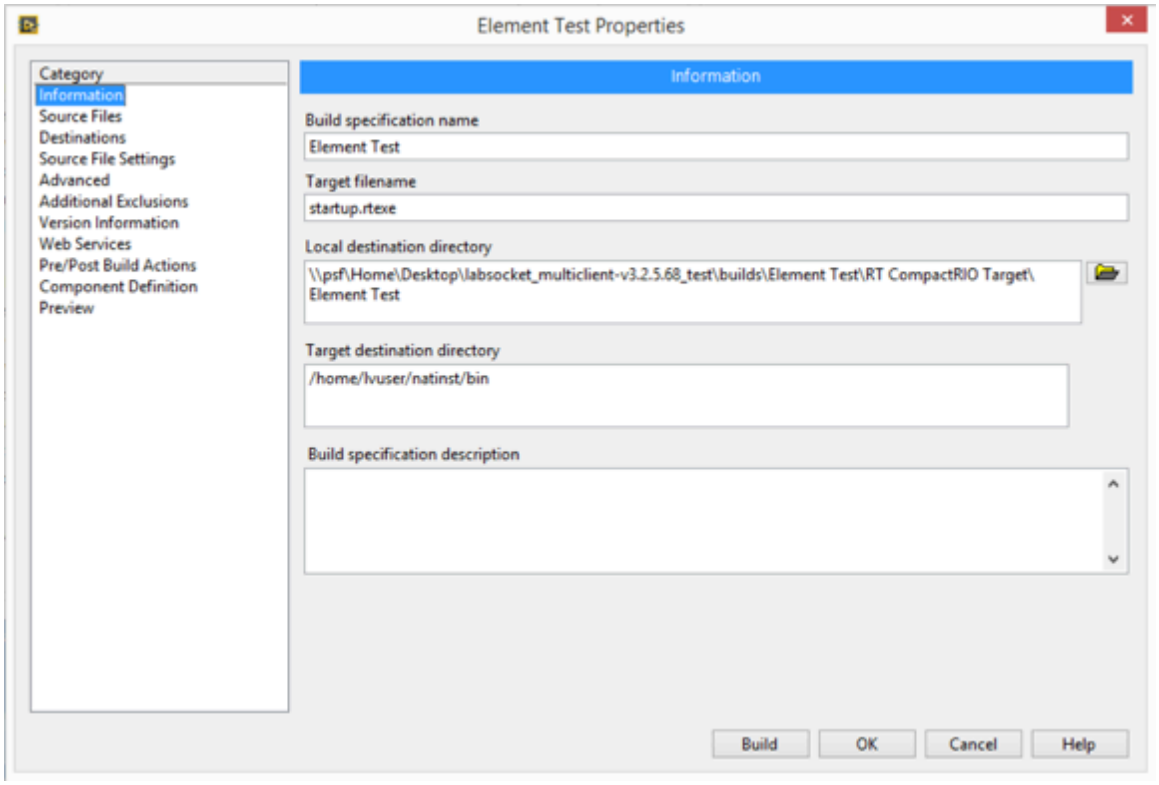


Figure 6.16. Real-Time Application Information Page

4. Select the **Source Files** page. Select the name of the Target VI ("LabSocket Demo – Element Test.vi") in this case and select it as a **Startup VI** (Figure 6.17).

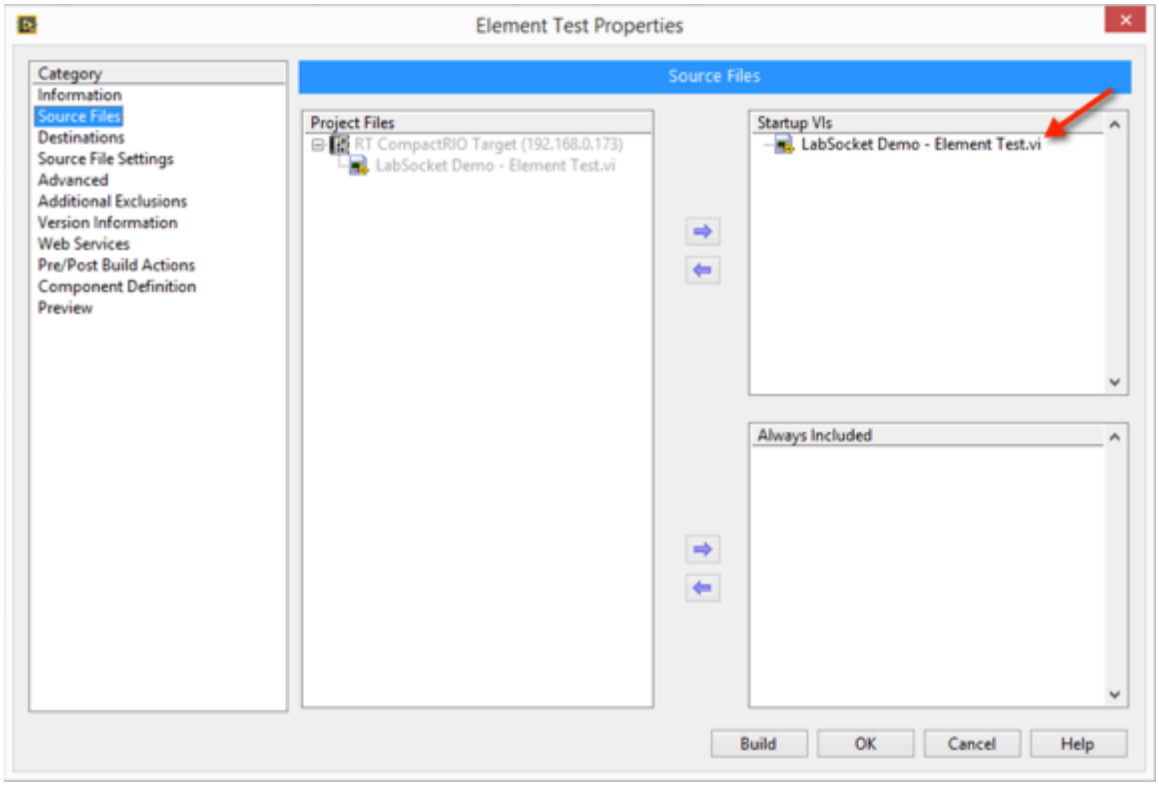


Figure 6.17. Real-Time Application Source Files Page

5. The default settings for the **Destinations** and **Source File Setting** categories are shown in Figures 6.18 and 6.19. These settings do not need to be changed.

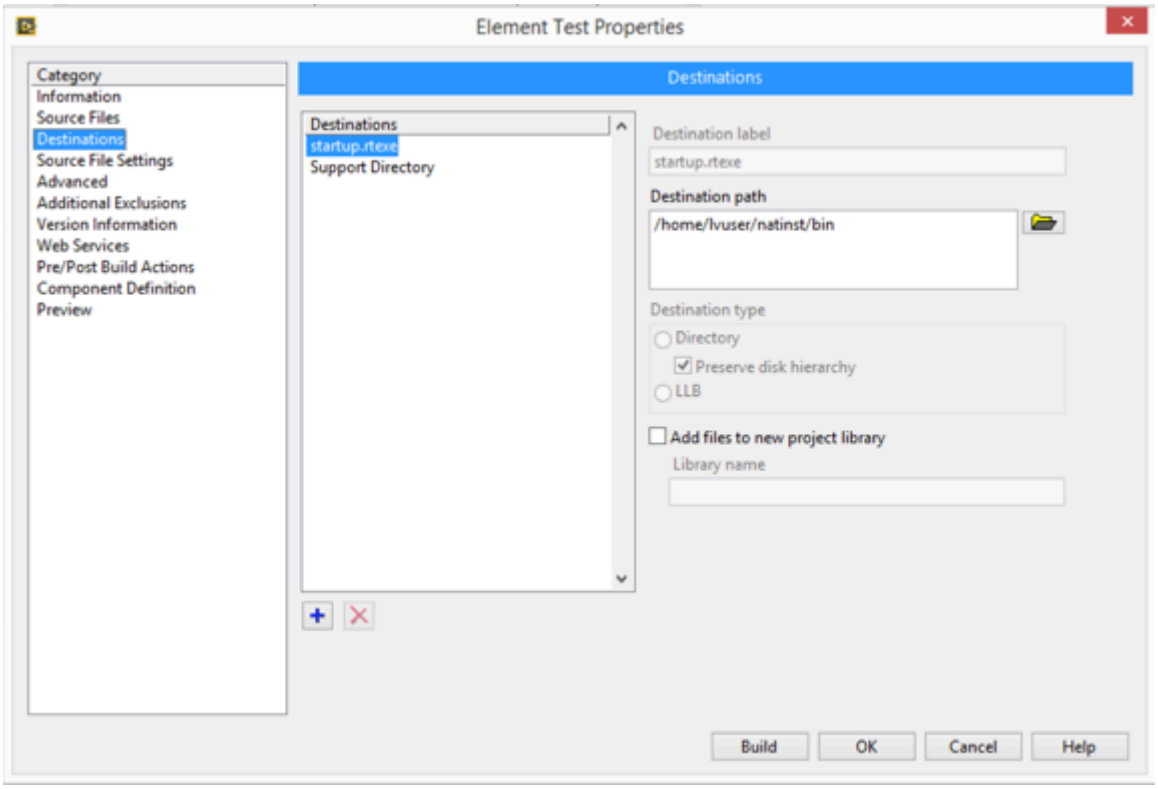


Figure 6.18. Real-Time Application Destinations Page

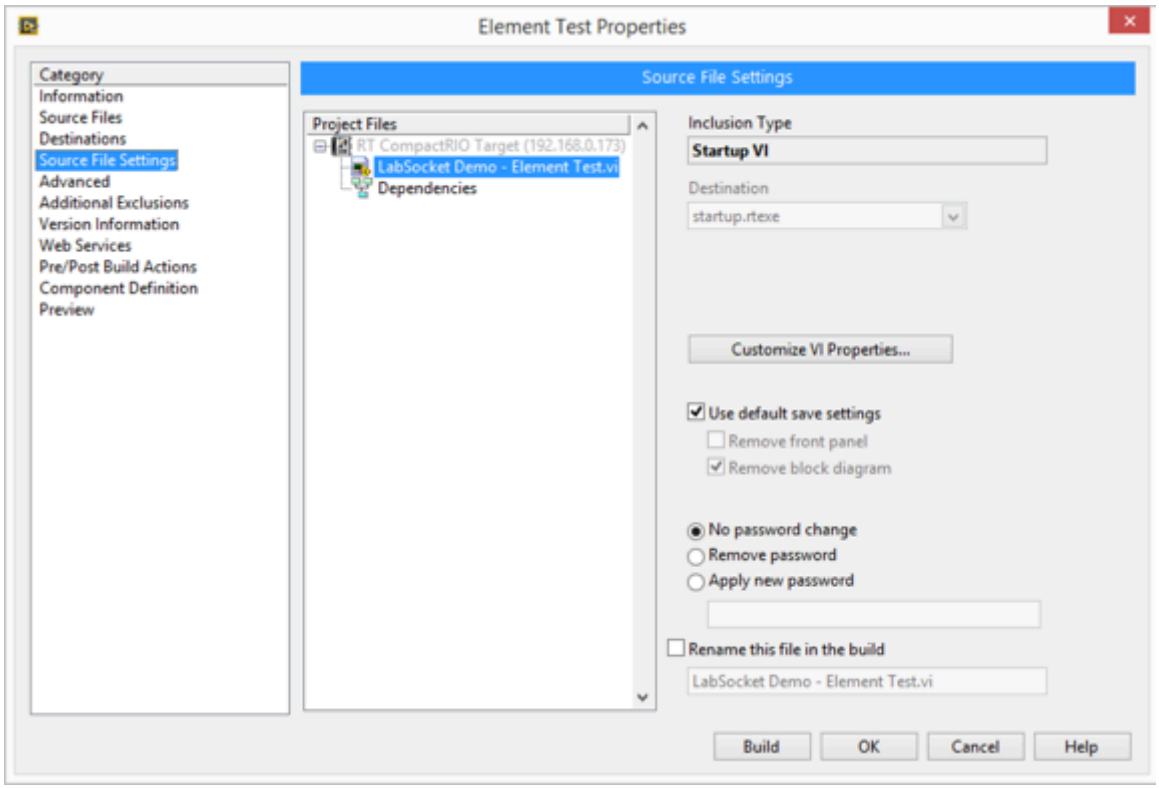


Figure 6.19. Real-Time Application Source File Settings Page

6. Click **Build** to build the real-time application.

7. Click **OK** to close the Real-Time Application properties window.

8. In the Project tree, right-click on the **build specification** ("Element Test" in this case). Select **Run as startup**.

9. Right-click on the **build specification.** Select **Deploy**.

10. Right-click on the target in the Project tree and select **Utilities > Restart** to restart the cRIO.

11. After the cRIO restarts, the Target VI may be accessed remotely in a browser using the URL displayed in the LabSocket Status Window shown on the display of the cRIO. Figure 6.20 demonstrates browser access to the "LabSocket Demo - Element Test.vi" operating on a cRIO-9030.

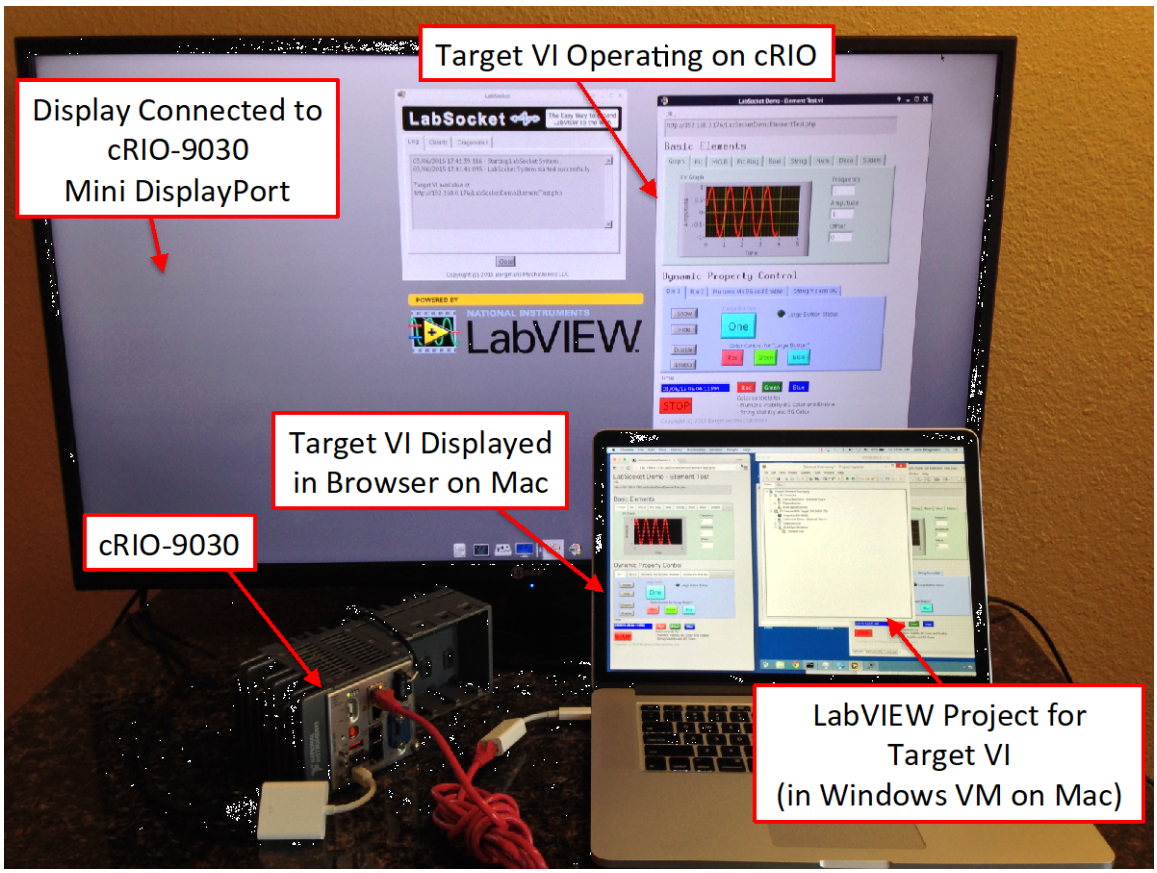


Figure 6.20. Browser Access to LabSocket Demo – Element Test.vi *operating on CompactRIO cRIO-9030*

7. Example Files

The LabSocket VIPM package installation process (Section 4.2, *LabSocket Software Installation)* automatically installs a set of example Target VIs that illustrate the capabilities the system. These VIs may be accessed by selecting **Help > Find Examples...** from the main LabVIEW window. Select the **Directory Structure** radio button. In the directory structure, double-click on the **Bergmans Mechatronics** directory and double-click on the **LabSocket** directory (Figure 7.1).

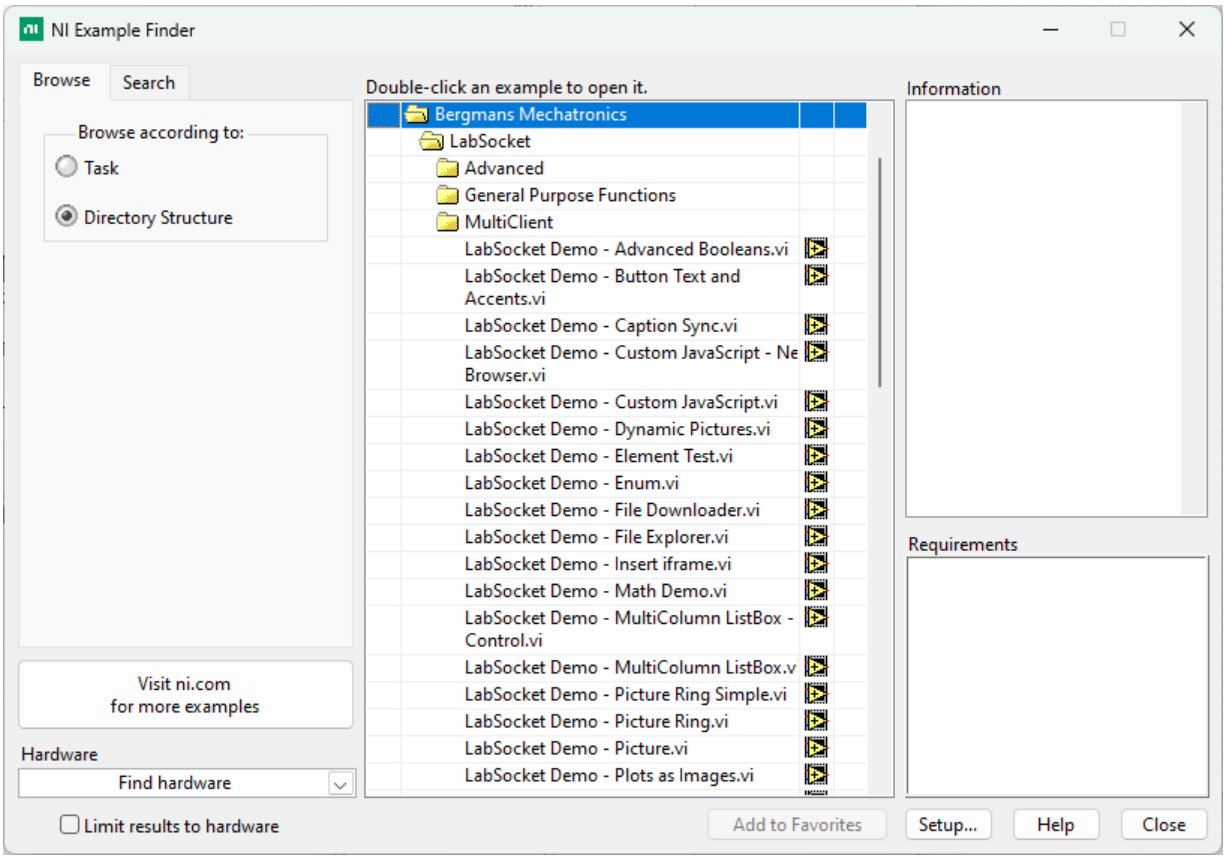


Figure 7.1 LabSocket Example Files

Three types of example VIs are included with the LabSocket VIPM:

- VIs demonstrating replication of front panel elements (Table 7.1). Although each of these VIs operates in Basic client mapping mode, the concepts demonstrated in these VIs are applicable to MultiClient mapping mode.

- VIs demonstrating advanced UI capabilities (Table 7.2) are included in the **Advanced** directory in the **LabSocket** example directory.

- VIs demonstrating MultiClient mapping (Table 7.3). These VIs can be found within the **MultiClient** directory in the **LabSocket** example directory.

Table 7.1. Front Panel Replication Example VIs

Filename	Description
LabSocket Demo - Advanced Booleans.vi	Demonstrates that booleans defined as Strict Type Defs are replicated using images of the true and false states
LabSocket Demo - Button Text and Accents.vi	Demonstrates support for accented characters and dynamic synchronization of boolean button text
LabSocket Demo - Caption Sync.vi	Demonstrates synchronization of caption text using "LabSocket Caption Sync.vi"
LabSocket Demo - Custom JavaScript.vi	Uses "LabSocket Custom JavaScript.vi" to insert arbitrary text into the browser client
LabSocket Demo - Custom JavaScript - New Browser.vi	Uses "LabSocket Custom JavaScript.vi" to insert into the browser client event handlers that open new browser windows
LabSocket Demo - Dynamic Pictures.vi	Demonstrates dynamic synchronization of pictures elements with those in the browser
LabSocket Demo - Element Test.vi	Demonstrates synchronization of Front Panel elements values with elements in browser. Elements include an XY Graph (replicated as an image), Picture element, MultiColumn ListBox and Tabs. Also demonstrates support for dynamic replication in the browser of Front Panel element properties such as: i) button visibility, color, text, and disabled state; ii) numeric control and indicator visibility; iii) numeric control disabled state; and, iv) string control and indicator visibility and background color.
LabSocket Demo - Enum.vi	Demonstrates synchronization of enumerated types
LabSocket Demo - File Downloader.vi	Uses "LabSocket Custom JavaScript.vi" to configure the browser client to download files from a cloud server
LabSocket Demo - File Explorer.vi	Illustrates how to use a MultiColumn ListBox to enable a remote user to select a file from a list of files on the Target VI platform
LabSocket Demo - Insert iframe.vi	Uses "LabSocket Custom JavaScript.vi" to insert a web page into an HTML iframe in the browser client
LabSocket Demo - Math Demo.vi	Illustrates how to use LabSocket in a multiple-VI system. The front panel of the main Target VI is accessible over the web while other VIs display the Target VI URL and a Stop button for the application
LabSocket Demo - MultiColumn ListBox - Control.vi	Demonstrates the use of a MultiColumn ListBox as a control element
LabSocket Demo - MultiColumn ListBox.vi	Demonstrates the eight combinations of MultiColumn ListBoxes that are possible based on i) row visibility; ii) column header visibility; and, iii) presence of vertical scrollbar.

Table 7.1 (Continued). Front Panel Replication Example VIs

Filename	Description
LabSocket Demo - Picture Ring Simple.vi	A simple demonstration of picture rings in action
LabSocket Demo - Picture Ring.vi	Picture ring demonstration
LabSocket Demo - Picture.vi	Demonstrates replication of a picture element in the browser
LabSocket Demo - Plots as Images.vi	Illustrates the use of the #LS_image preprocessor tag to enable replication of plots as images
LabSocket Demo - Preprocessor.vi	Illustrates the use of the #LS_no_display and #LS_no_sync preprocessor tags to hide and unsynchronize, respectively, front panel elements
LabSocket Demo - Stub VI.vi	Demonstrates the use of the "element stub.vi" that enables developers to create custom JavaScript code for unsupported front panel elements.
LabSocket Demo - Tabs.vi	Demonstates synchornization of tab element pages
LabSocket Demo - Waveform Chart.vi	Demonstration of support for Waveform Charts
LabSocket Demo - Waveform Graph.vi	Demonstration of support for Waveform Graphs
LabSocket Demo - XY Graph.vi	Demonstration of support for XY Graphs
Simple Demo.vi	Simple VI to demonstrate LabSocket operation

Table 7.2. Advanced UI Example Vis

Filename	Description
Advanced UI.vi	Demonstrates advanced UI capabilities, including: - slider controls represented as images - control of cursors and plot visibilty for XY graphs represented as images - unit synchronization for numeric controls and indicators
Advanced UI - MultiClient.vi	VI for operation of Advanced UI.vi in MultiClient mapping mode.
Table Support.vi	Demonstrates support for table controls and indicators.

Table 7.3. MultiClient Mapping Example VIs

Filename	Description
Quiz Server 1.vi	Simple demonstration of MultiClient mapping. Uses "Quiz 1.vi" as Target VI.
Quiz Server 2.vi	MultiClient mapping demonstration that includes interprocess communication with Target VI instances. Uses Basic mapping for browser-based access to top-level VI. Uses "Quiz 2.vi" as Target VI.

8. LabSocket Start VIs

The VIs ***LabSocket Start.vi*** and ***LabSocket-MC Start.vi*** are used to launch the routines that perform the Target VI Front Panel screenscrape and synchronize the front panel and browser.

In Basic client mapping applications, the VI ***LabSocket Start.vi*** (Figure 8.1) is placed on the block diagram on the Target VI. For MultiClient client mapping applications, the VI ***LabSocket-MC Start.vi*** (Figure 8.2) is placed on block diagram of the top-level VI of the application.

Connections to these two VIs are summarized in Table 8.1.

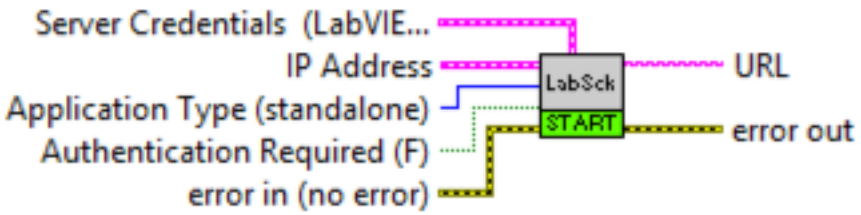


Figure 8.1. LabSocket Start VI

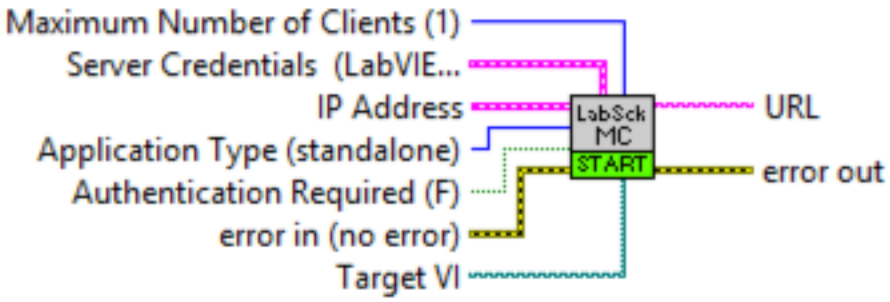


Figure 8.2. LabSocket-MC Start VI

Table 8.1. Terminals of LabSocket Start.vi *and* LabSocket-MC Start.vi

Terminal	Description	Default Input Value
Maximum Number of Clients	Defines the maximum number of clients that can connect to the system. (*LabSocket-MC Start.vi* only)	1
Server Credentials	Cluster containing username and password for access to LabSocket Server	LabVIEW / LabVIEW
IP Address	Cluster containing two strings: • *LabSocket_Server* – IP Address of LabSocket Server relative to LabVIEW Host platform. Section 4.1, Step 9 describes how to determine this value. • *External* – IP Address of LabSocket Server relative to browser. Leave blank if same as *LabSocket_Server* IP address. To use the LabSocket server software on the cloud server at labsocket.com during initial evaluation testing, set *LabSocket_Server* to "labsocket.com" (no quotes) and leave *External* blank	XX.XX.XX.XX / blank
Application Type	• standalone – Target VI is rendered alone in a web page • moodle_integration – Target VI is rendered within an iFrame in a Moodle course web page. Contact Bergmans Mechatronics for more information on this option.	standalone
Authentication Required	User authentication requirement for browser access • True – A pop-up window appears in browser when web page loads. User must enter credentials before web page becomes active (Figure 8.3) • False – User credentials are not requested. (In this case, the browser sends the credentials "client / client" to the message broker. Unless these credential settings have been changed, browser access will always be permitted.) See Section 11, *User Administration*, for additional details about user administration and authentication.	False
error in	Errors that occur before *LabSocket Start(-MC).vi* is called	no error
Target VI	Path to Target VI. (*LabSocket-MC Start.vi* only)	empty path
error out	Errors that occur prior to calling or within *LabSocket Start(-MC).vi*	*NA*
URL	URL for access to Target VI	*NA*

Figure 8.3. Login Window That Appears When Authentication Required *Input is True*

9. LabSocket Status Window

The LabSocket Status Window provides details on the operation of the LabSocket software. The window appears when either the ***LabSocket Start.vi*** or ***LabSocket-MC Start.vi*** are called and contains three pages: 1) a Log page; 2) a Clients page; and, 3) a Diagnostics page. Each of these pages is described in detail below.

Note that calls to either ***LabSocket Start.vi*** and ***LabSocket-MC Start.vi*** will launch the Status Window only if the window is not already active.

9.1 Log Page

The URL at which the Target VI may be accessed will be displayed in the Log page of this window (Figure 9.1). This URL will match that output on the URL terminal of ***LabSocket(-MC) Start.vi***.

Figure 9.1. LabSocket Status Window (Log Page)

9.2 Clients Page

The Clients page (Figure 9.2) displays the Client ID of each connected browser in the "Browser ID" column. The "Last Heartbeat Echo" column displays the time of the last response from the browser to periodic test signals that are transmitted to each connected browser. (If an echo signal is not returned from a browser in greater than 10 seconds, the client is considered inactive and removed from the list of clients in the Status window.)

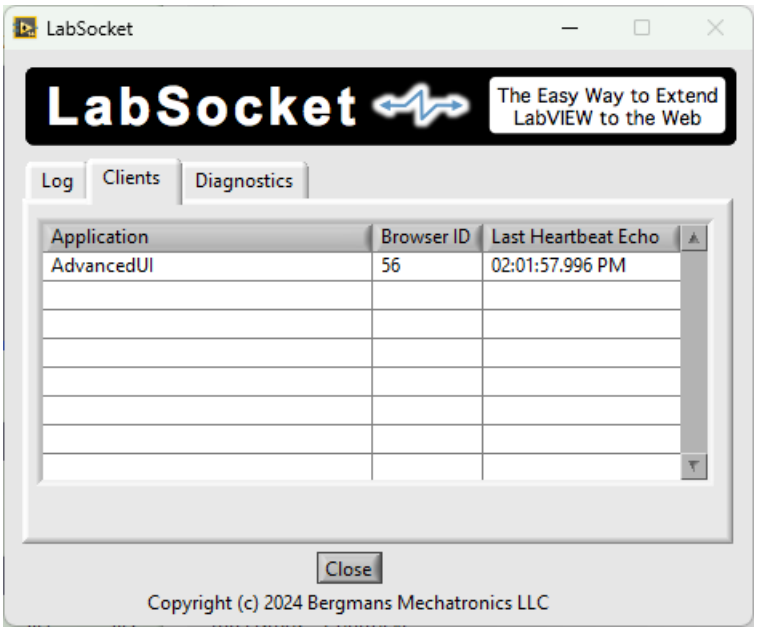


Figure 9.2. LabSocket Status Window (Clients Page)

9.3 Diagnostics Page

The Diagnostics page (Figure 9.3) of the **LabSocket Status** window displays a "Loop Period" value. This value is the period of time required to update the front panel of the Target VI with any changes sent from the browser plus the time required to detect changes in the Target VI front panel and to send these changes to each connected browser client.

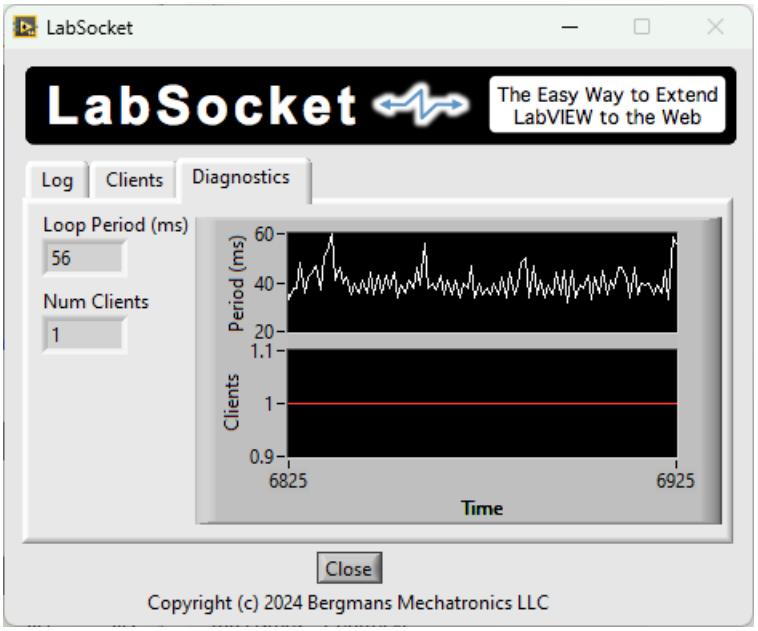


Figure 9.3. LabSocket Status Window (Diagnostics Page)

The Diagnostics page also includes a display of the number of clients currently connected to the Target VI

Click **Close** to close the Status window and discontinue the synchronization of the Target VI and browser. Closing the Status window will not stop the Target VI and stopping the Target VI does not close the Status window.

10. Advanced Features

This section provides details on the following advanced features of LabSocket:

- preprocessor tags that allow fine tuning of the browser client;
- picture ring support and high-fidelity replication of boolean elements;
- insertion of custom JavaScript code into the browser client; and,
- replication of PNG images in the browser
- system parameter initialization
- system monitoring
- caption synchronization
- detailed diagnostics data output

10.1 Preprocessor Tags

Preprocessor tags in the caption or label text of front panel may be used to specify how individual elements are to be replicated in the browser. These tags are summarized in Table 10.1.

Table 10.1. Preprocessor Tags

Tag	Description	Example Application
#LS_no_display	Do not replicate the target element in the browser	Prevent display of a Target VI stop button
#LS_no_sync	Prevents synchronization of the front panel element with its browser representation	Prevent tab element synchronization in a Basic mapping application
#LS_image(*compression* ,*decimation*)	Display the target element as an image in the browser. Updated image is only sent to browser if image change detected. Optional arguments are: i) *compression* - image compression (0=no image compression, 9=maximum image compression); and, ii) *decimation* - number of iterations of main LabSocket processing loop to skip before checking for image update. Not supported on cRIO-903x targets.	Display an exact replica of a XY Graph (including cursors and graph visibility selection via legend) and sliders (including coercion and pop-up display of current value). See example VI "LabSocket Demo - Advanced UI.vi"

The effect of the #LS_no_display tag is shown in the screenshot of the example VI "LabSocket Demo - Preprocessor.vi" in Figure 10.1. The example VI "LabSocket Demo - Plots as Images.vi" also demonstrates the effect of preprocessor tags (see Section 7, *Example VIs*).

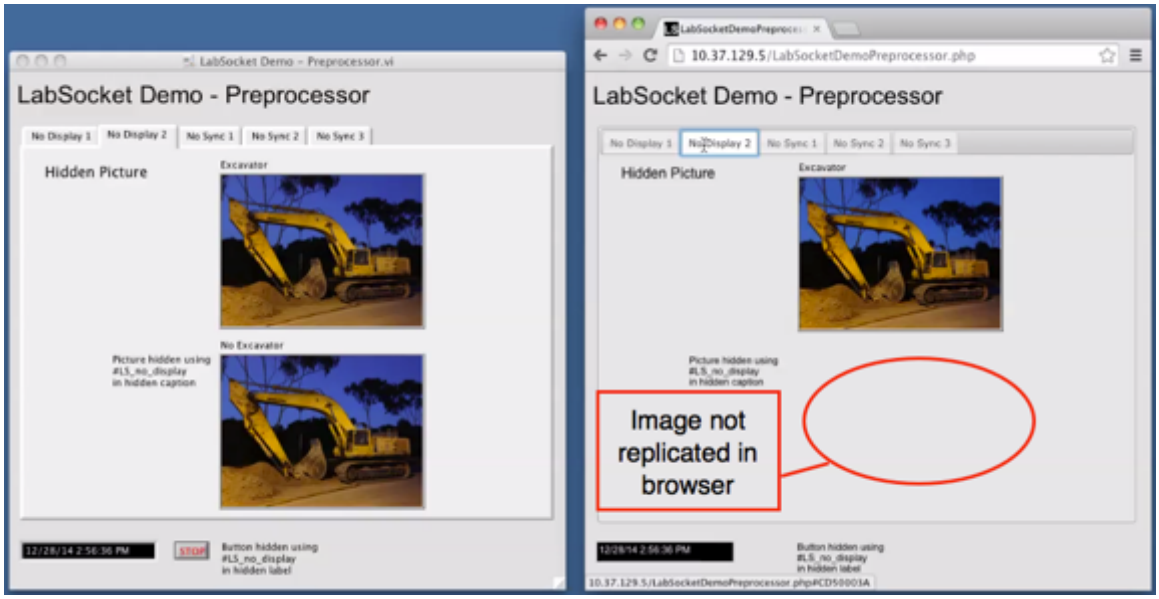


Figure 10.1. Screenshot of "LabSocket Demo – Preprocessor.vi" *demonstrating the use of "*#LS_no_display" *preprocessor tag to prevent replication of a picture (L- LabVIEW VI, R – Browser)*

10.2 Picture Rings and High-Fidelity Boolean Elements

LabSocket supports replication of Picture Ring controls and indicators. This replication only occurs if the element is a type def, otherwise the element is not replicated. The example VI "LabSocket Demo – Picture Ring Simple.vi" illustrates Picture Ring control and indicator replication (Figure 10.2).

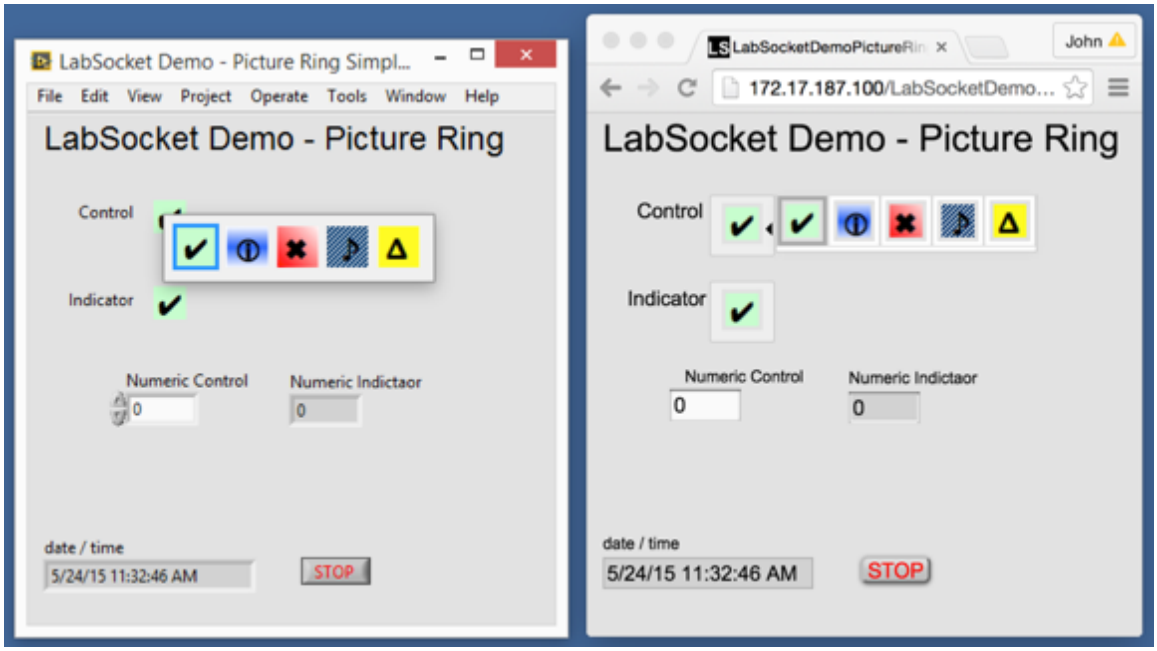


Figure 10.2. Screenshot of "LabSocket Demo – Picture Ring Simple.vi" *demonstrating the replication of a picture ring control in the browser (L- LabVIEW VI, R – Browser)*

LabSocket also supports high-fidelity replication of Boolean controls and element if the element is a type def. A screenshot of the example VI "LabSocket Demo – Advanced Booleans.vi" and its browser representation is illustrated in Figure 10.3.

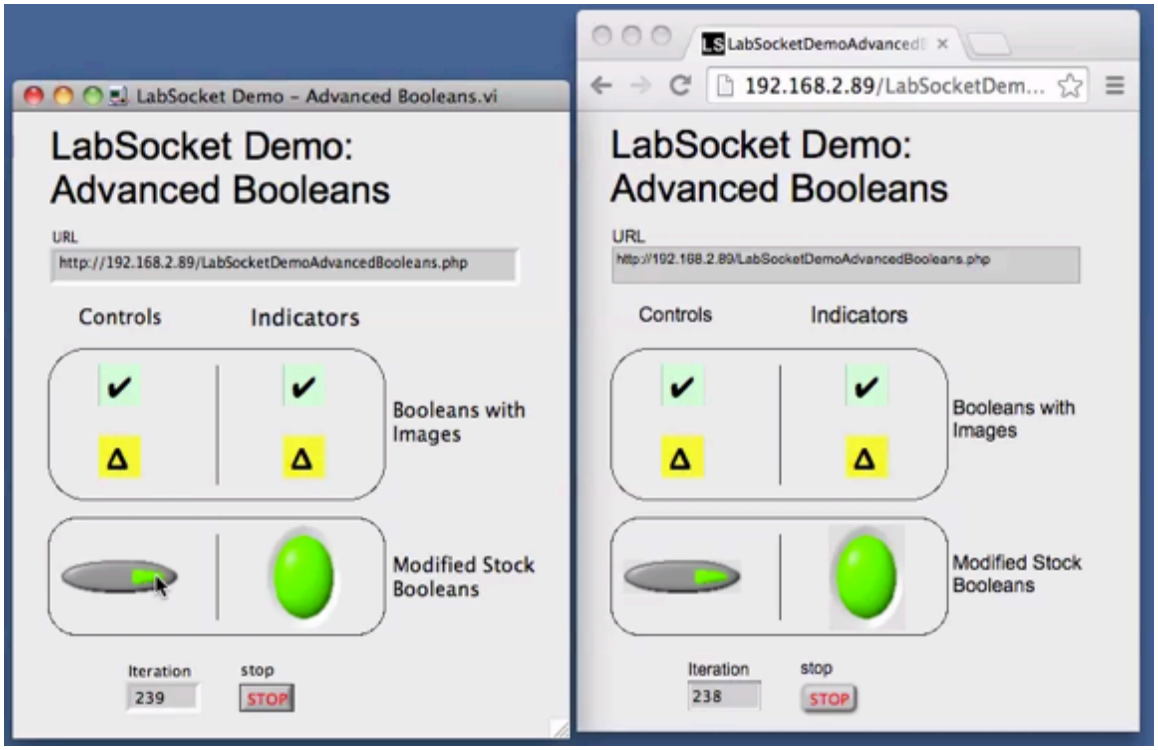


Figure 10.3. Screenshot of "LabSocket Demo – Advanced Booleans.vi" *demonstrating high-fidelity representation of Boolean elements (L- LabVIEW VI, R – Browser)*

10.3 Developer Customization

LabSocket provides developers with the ability to customize the browser client software using two options: i) adding arbitrary JavaScript to the browser client; and, ii) adding custom JavaScript code for currently unsupported front panel element types.

10.3.1 Arbitrary JavaScript for the Browser

The "LabSocket Custom JavaScript.vi" VI allows arbitrary JavaScript code to be included in the browser client. This VI is located in the tools palette at **Bergmans Mechatronics > LabSocket > Advanced > LabSocket Custom JavaScript .vi**. Any text wired to the "JavaScript In" terminal of the VI will be included in the browser client code and executed when the browser page loads. This VI must be called before the "LabSocket Start.vi".

The example VI "LabSocket Demo – Custom JavaScript.vi" illustrates the use of this VI to display custom text in an html element in the browser client. A screenshot of the block diagram of this VI is shown in Figure 10.4. The front panel and browser representation are shown in Figure 10.5.

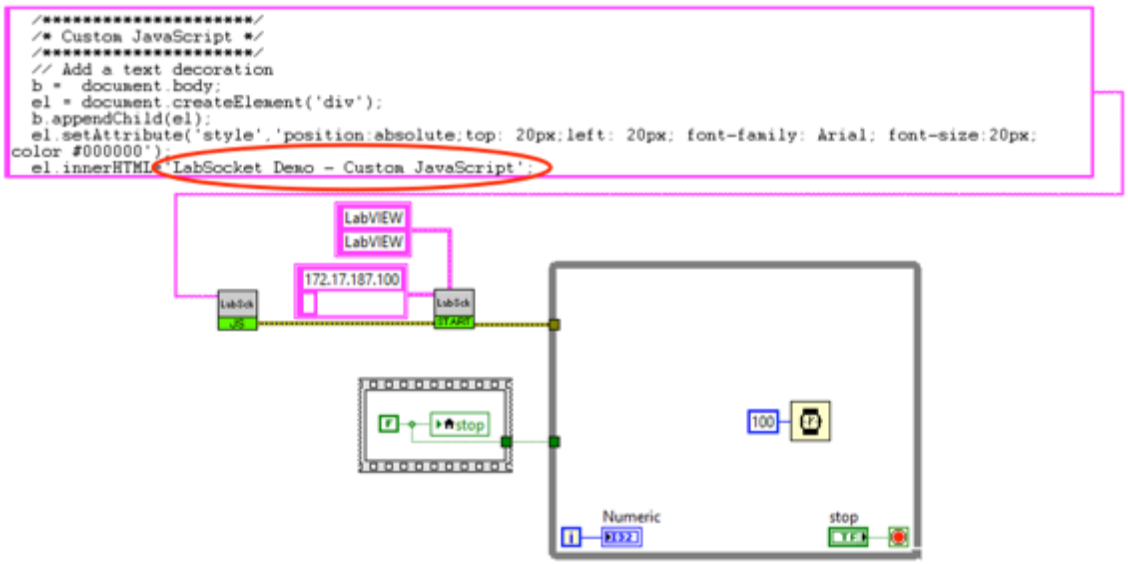


Figure 10.4. Screenshot of block diagram of "LabSocket Demo – Custom JavaScript.vi"

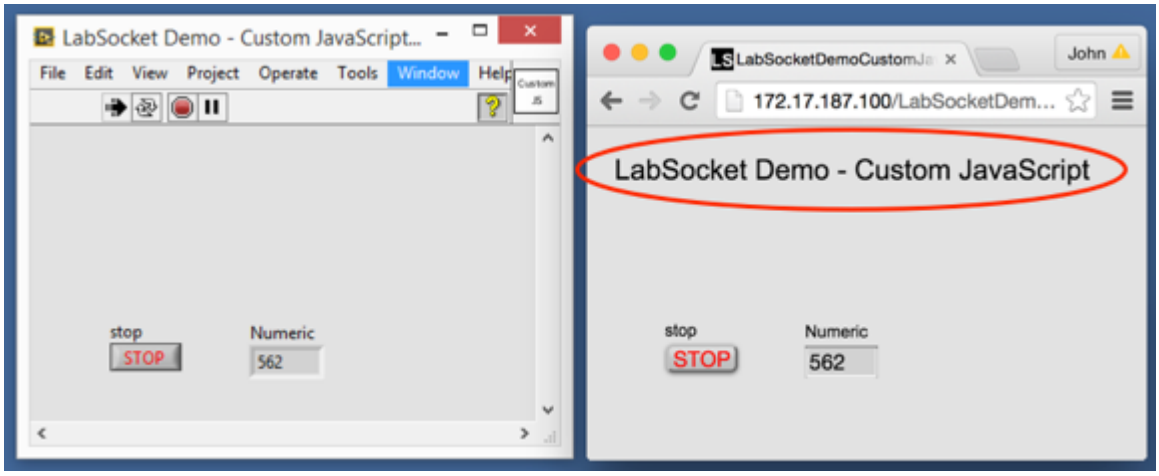


Figure 10.5. Screenshot of "LabSocket Demo – Custom JavaScript.vi" *illustrating the use of arbitrary JavaScript code to customize the browser client (L- LabVIEW VI, R – Browser)*

10.3.2 Custom JavaScript for Unsupported Elements

The second customization option is to add arbitrary JavaScript code to the browser for LabVIEW front panel element types that are not supported by the LabSocket system.

During the initial web client creation process, if LabSocket encounters an element on the Target VI front panel with a VI Server Class Name that is not supported, it calls a VI named "element stub.vi". Developers may make any desired modifications to that VI to add their own JavaScript code to the browser for these unsupported elements. To assist with the customization, the VI Server reference to the specific unsupported element is passed into "element stub.vi".

The VI "element stub.vi" is located in the directory
"<vi.lib>/LabSocket/developer.lib/vi.lib/CreateJavaScript" where <vi.lib> is typically
"C:/Program Files/National Instruments/LabVIEW 20XX/vi.lib"

As an example of how this VI might be used, the "element stub.vi" currently contains simple JavaScript code that adds html elements at the locations of arrays or clusters on the VI front panel. Figure 10.6 shows the case for handling array elements. The example VI "LabSocket Demo – Stub.vi" illustrates this capability (Figure 10.7).

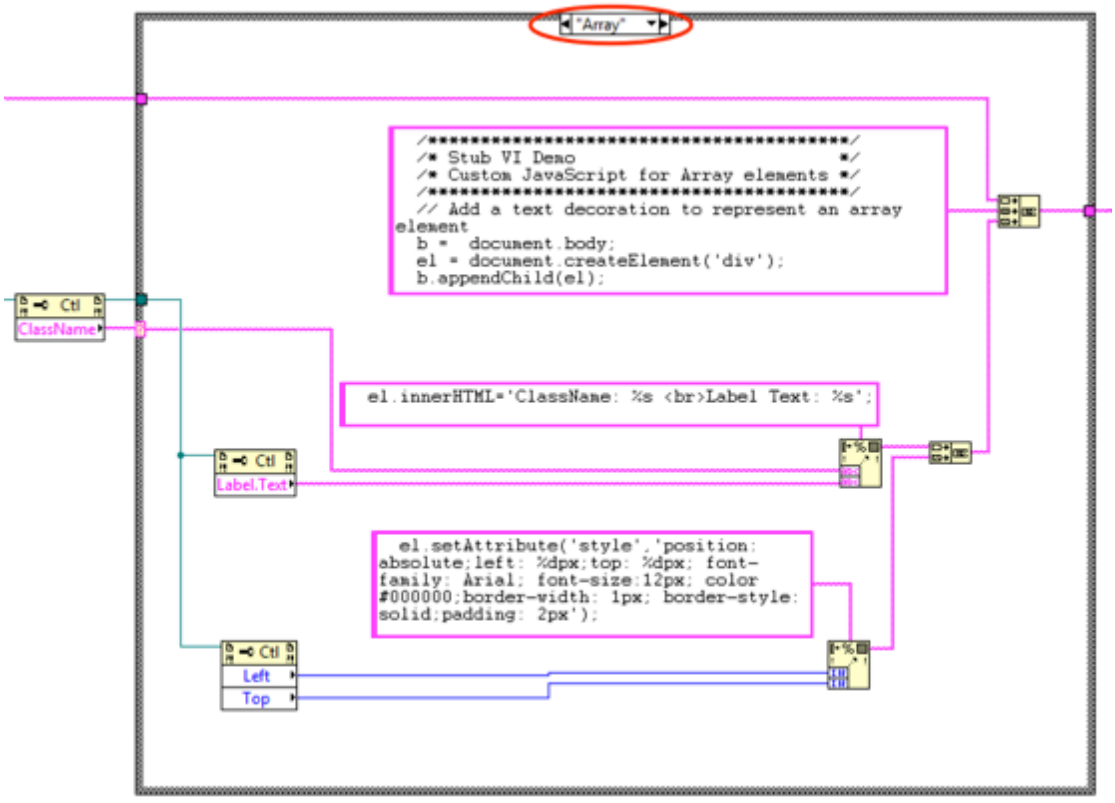


Figure 10.6. Portion of Block Diagram of "element stub.vi"

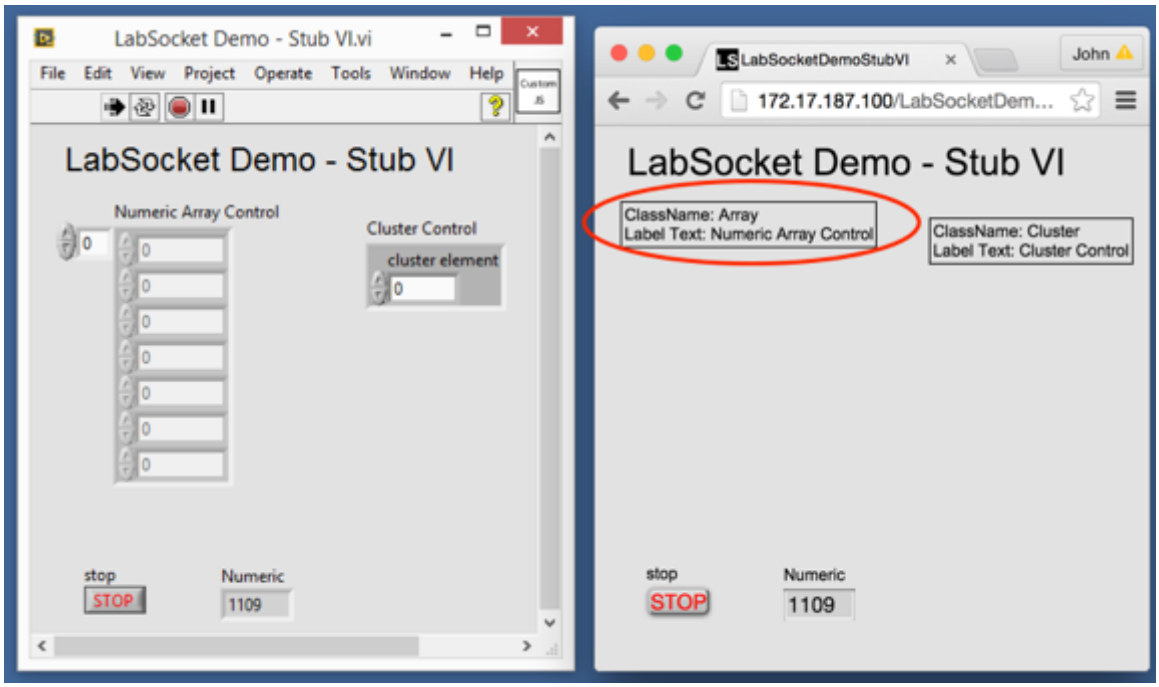


Figure 10.7. Screenshot of "LabSocket Demo – Stub VI.vi" *illustrating the use* of "element – stub vi" *to add custom browser representations for currently unsupported front panel elements (L- LabVIEW VI, R – Browser)*

10.4 Working with Images

A "PNG to Picture Converter" tool is provided with the LabSocket system to enable images contained in PNG files to be easily rendered in the browser. The process for working with images is as follows.

1. From the menu bar, select
   Tools> Bergmans Mechatronics > LabSocket > PNG to Picture Converter…

2. Select the PNG file of interest.

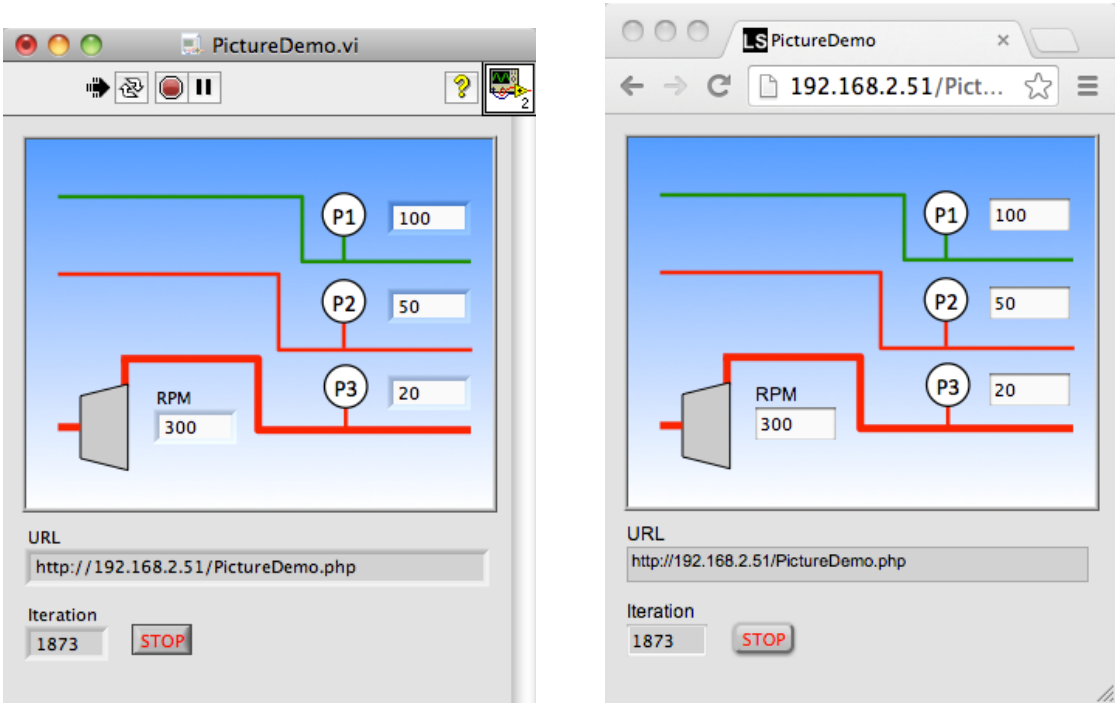
3. A new, unnamed VI will appear. Copy the picture control from this VI to the front panel of the Target VI.

4. Right-click on the picture. Select **Data Operations>Make Current Value Default**

5. Save the Target VI

6. Run the Target VI (assuming the ***LabSocket Start.vi*** and other components necessary for a practical application are already present on the VI block diagram).

A simple Target VI using a picture element and its representation in a browser is shown in Figure 10.8.



a) Front Panel *b) Browser Representation*

Figure 10.8. PictureDemo.vi *Illustrating Rendering of Picture Elements in Browser*

Controls and indicators may be superimposed in front of the picture element. Note that elements in the browser are displayed from back to front in order of increasing tabbing order value (ie tabbing order value 0 is at the back of the browser representation, with other elements displayed in front of it). Similarly, for elements in tab pages, elements are displayed from front to back in order of increasing tab "Control Order".

10.5 System Parameter Initialization

Although the default LabSocket system parameters are sufficient for most applications, certain system parameters may be initialized, if necessary, to different values using the LabSocket Init VI (Figure 10.9).

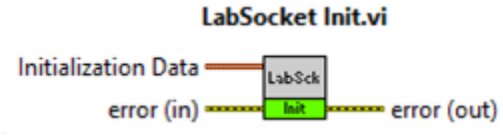


Figure 10.9. LabSocket Init.vi

The Initialization Data input cluster contains the following parameters:

- wait time (ms) - Wait period, in milliseconds, in the main LabSocket processing loop. Default value: 20 ms
- heartbeat_period_sec - Period, in seconds, between heartbeat message transmissions to browser client(s). Browser clients normally return a heartbeat reply message in response. Default value: 2 sec.
- broker_keepalive_period_sec - Period, in seconds, between message transmissions to message broker to keep connection between Target VI and broker active. Default value: 15 sec.
- connection_timeout_sec - Maximum period, in seconds, to wait for a heartbeat reply message from a browser client. If reply is not received in this period, client is assumed to have disconnected from Target VI. Default value: 10 sec.

Note that this VI must be called before LabSocket Start.vi or LabSocket-MC Start.vi.

This VI is located in the tools palette at
Bergmans Mechatronics > LabSocket > Advanced > LabSocket Init.vi.

10.6 System Monitoring

The LabSocket system may be monitored using the LabSocket Monitor VI (Figure 10.10). This VI provides status information on the performance of the main software loop, known as the LabSocket Primary Processing Loop.

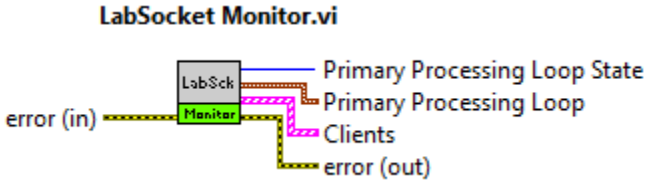


Figure 10.10. LabSocket Monitor.vi

For best results, the Target VI and the optional initialization VI "LabSocket Init.vi" should be configured such that the Primary Processing Loop Period is less than 500 ms.

This VI also provides information about each client that is connected to the system.

The example VI "Advanced UI.vi", located in the Example directory Advanced \ Advanced UI, illustrates the use of this VI.

The LabSocket Monitor VI is located in the tools palette at
Bergmans Mechatronics > LabSocket > Advanced > LabSocket Monitor.vi.

10.8 Caption Synchronization

Some applications require dynamic synchronization of captions on the VI front panel. This capability can be enabled by placing the LabSocket Caption Sync VI (Figure 10.11) on the block diagram and wiring a True constant to the Caption Sync input. This VI also synchronizes the page names of tab elements.

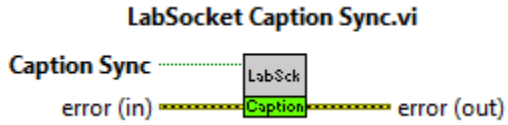


Figure 10.11. LabSocket Caption Sync.vi

Note that this VI must be called before LabSocket Start.vi or LabSocket-MC Start.vi.

The example VI ""LabSocket Demo - Caption Sync.vi" included with LabSocket illustrates this capability. The LabSocket Caption Sync VI is located in the tools palette at
Bergmans Mechatronics > LabSocket > Advanced > LabSocket Caption Sync.vi.

10.9 Detailed Diagnostics Output

The LabSocket Diagnostics VI (Figure 10.12) enables output of detailed information about LabSocket operation to a text file.

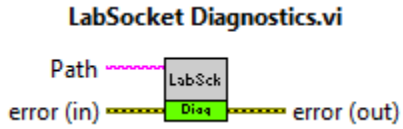


Figure 10.12. LabSocket Diagnotics.vi

This VI must be called before LabSocket Start.vi or LabSocket-MC Start.vi.

The LabSocket Diagnostics VI is located in the tools palette at
Bergmans Mechatronics > LabSocket > Advanced > LabSocket Diagnotics.vi.

11. User Administration

All applications and users connecting to the ActiveMQ message broker require proper credentials before transmitting or receiving data via the broker. By default, the broker loads credentials from a configuration file on startup. This section describes how to modify the credentials in this configuration file.

LDAP User Authentication Option

The message broker can optionally be configured to validate user credentials stored in a Lightweight Directory Access Protocol (LDAP) server. Benefits of the use of an LDAP server are:

- users can be authenticated using credentials in a customer's existing LDAP database
- administrators can manage users without restarting the broker
- a browser-based interface such as phpLDAPadmin can be used to manage user data: (http://phpldapadmin.sourceforge.net/wiki/index.php/Main_Page)

Contact Bergmans Mechatronics for details on implementing LDAP user management or to request a VM containing an OpenLDAP server.

In the instructions below, two locations within the VM directory structure are defined as follows:

```
$LABSOCKET_HOME = /home/labsocket
$ACTIVEMQ_HOME = /home/labsocket/apache-activemq-5.9.0
```

To modify the credentials in the ActiveMQ configuration file:

1. Log in to LabSocket Server VM using an ssh client or the VirtualBox terminal window.
2. Navigate to `$ACTIVEMQ_HOME/conf`
3. Open **activemq-LabSocket.xml** in the vi editor
4. The `<simpleAuthenticationPlugIn>` section beginning on line 47 defines the username and passwords for each user that is permitted to access the ActiveMQ message broker.

   The default users are described in Table 11.1.

Table 11.1. Default User Definitions of ActiveMQ Message Broker in LabSocket Server VM

Username	Description
LabVIEW	name used by LabSocket Support VIs. This username and its associated password are wired to the ***Server Credentials*** input of ***LabSocket Start.vi***
LabSocketPHP	name used by PHP-based administrative program. Credentials for this client are defined in the file `$LABSOCKET_HOME/LabSocket_php/credentials.txt`
client	remote browser user
client2	remote browser user

5. To add, delete or change user credentials, edit the settings in the `<simpleAuthenticationPlugIn>` section as required. If changing the credentials for the LabSocket Support VIs or PHP Administrative program, but sure to change the credential settings supplied by these clients (ie. Inputs to ***LabSocket Start.vi*** or contents of `$LABSOCKET_HOME/LabSocket_php/credentials.txt`.)

6. Save and exit **activemq-LabSocket.xml.**

To activate the new user settings, restart the Virtual Machine using the command

```
sudo shutdown –r now
```

Enter the password for the user `labsocket` if prompted.