Contents

Student Guide
A. Course Description .................................................................................................. vi
B. What You Need to Get Started ............................................................................... vi
C. Installing the Course Software ............................................................................... vii
D. Course Goals ........................................................................................................ vii
E. Course Conventions ............................................................................................... viii

Lesson 1
Common Design Techniques
A. Single Loop Architectures .................................................................................... 1-2
B. Parallelism ............................................................................................................. 1-6
C. Multiple Loop Architectures .................................................................................. 1-8
D. Timing a Design Pattern ....................................................................................... 1-12

Lesson 2
Communicating Among Multiple Loops
A. Variables ............................................................................................................... 2-2
B. Functional Global Variables .................................................................................. 2-12
Exercise 2-1 Variables VI ....................................................................................... 2-15
C. Race Conditions ................................................................................................... 2-24
Exercise 2-2 Concept: Bank VI ................................................................................ 2-31
D. Synchronizing Data Transfer .............................................................................. 2-34
Exercise 2-3 Queues versus Local Variables VI ..................................................... 2-39
Exercise 2-4 Optional: Global Data Project ............................................................ 2-44

Lesson 3
Improving an Existing VI
A. Refactoring Inherited Code .................................................................................. 3-2
B. Typical Issues ........................................................................................................ 3-4
Exercise 3-1 Concept: Typical Issues ...................................................................... 3-8

Lesson 4
Controlling the User Interface
A. VI Server Architecture .......................................................................................... 4-2
B. Property Nodes ..................................................................................................... 4-3
Exercise 4-1 Temperature Limit VI .......................................................................... 4-5
C. Control References .............................................................................................. 4-10
Exercise 4-2 Set Plot Names .................................................................................... 4-14
D. Invoke Nodes ....................................................................................................... 4-24
Exercise 4-3 Front Panel Properties VI ................................................................. 4-25
Contents

Lesson 5
Advanced File I/O Techniques
A. File Formats ..........................................................................................................5-2
B. Binary Files ..........................................................................................................5-5
Exercise 5-1 Bitmap File Writer VI ......................................................................5-12
C. TDM Files .............................................................................................................5-19
Exercise 5-2 TDM Query VI .................................................................................5-31

Lesson 6
Creating and Distributing Applications
A. LabVIEW Features for Project Development .......................................................6-2
Exercise 6-1 Concept: LabVIEW Project Management Tools ..............................6-5
B. Preparing the Application .....................................................................................6-7
C. Building the Application and Installer ...............................................................6-9
Exercise 6-2 Concept: Creating a Stand-Alone Application .................................6-11

Appendix A
Additional Information and Resources

Course Evaluation
When writing programs, often you must change the attributes of front panel objects programmatically. For example, you may want to make an object invisible until a certain point in the execution of the program. In LabVIEW, you can use VI Server to access the properties and methods of front panel objects. This lesson explains the VI Server, Property Nodes, control references, and Invoke Nodes.

**Topics**

A. VI Server Architecture  
B. Property Nodes  
C. Control References  
D. Invoke Nodes
A. VI Server Architecture

The VI Server is an object-oriented, platform-independent technology that provides programmatic access to LabVIEW and LabVIEW applications. VI Server performs many functions; however, this lesson concentrates on using the VI Server to control front panel objects and edit the properties of a VI and LabVIEW. To understand how to use VI Server, it is useful to understand the terminology associated with it.

Object-Oriented Terminology

Object-oriented programming is based on objects. An object is a member of a class. A class defines what an object is able to do, what operations it can perform (methods), and what properties it has, such as color, size, and so on.

Objects can have methods and properties. Methods perform an operation, such as reinitializing the object to its default value. Properties are the attributes of an object. The properties of an object could be its size, color, visibility, and so on.

Control Classes

LabVIEW front panel objects inherit properties and methods from a class. When you create a Stop control, it is an object of the Boolean class and has properties and methods associated with that class, as shown in Figure 4-1.

VI Class

Controls are not the only objects in LabVIEW to belong to a class. A VI belongs to the VI Class and has its own properties and methods associated with it. For instance, you can use VI class methods to abort a VI, to adjust the position of the front panel, and to get an image of the block diagram. You can use VI class properties to change the title of a front panel window, to retrieve the size of the block diagram, and to hide the Abort button.

Figure 4-1. Boolean Class Example
B. Property Nodes

Property Nodes access the properties of an object. In some applications, you might want to programmatically modify the appearance of front panel objects in response to certain inputs. For example, if a user enters an invalid password, you might want a red LED to start blinking. Another example is changing the color of a trace on a chart. When data points are above a certain value, you might want to show a red trace instead of a green one. Property Nodes allow you to make these modifications programmatically. You also can use Property Nodes to resize front panel objects, hide parts of the front panel, add cursors to graphs, and so on.

Property Nodes in LabVIEW are very powerful and have many uses. This section describes examples of specific properties that can change the appearance and function of front panel objects programmatically. Refer to the LabVIEW Help for more information about Property Nodes.

Creating Property Nodes

When you create a Property Node from a front panel object by right-clicking the object, selecting Create»Property Node, and selecting a property from the shortcut menu, LabVIEW creates a Property Node on the block diagram that is implicitly linked to the front panel object.

If the object has an owned label, the Property Node has the same label. You can change the label after creating the node. You also can create multiple Property Nodes for the same object.

Using Property Nodes

When you create a Property Node, it initially has one terminal representing a property you can modify for the corresponding front panel object. Using this terminal on the Property Node, you can either set (write) the property or get (read) the current state of that property.

For example, if you create a Property Node for a digital Numeric control using the Visible property, a small arrow appears on the right side of that terminal, indicating that you are reading that property value. You can change the action to write by right-clicking the terminal and selecting Change To Write from the shortcut menu. Wiring a Boolean False to the Visible property terminal causes the numeric control to vanish from the front panel when the Property Node receives the data. Wiring a Boolean True causes the control to reappear.
To get property information, right-click the node and select **Change to Read** from the shortcut menu. To set property information, right-click the node and select **Change to Write** from the shortcut menu. If the small direction arrow on the property is on the right, you are getting the property value. If the small direction arrow on a property is on the left, you are setting the property value. If the Property Node in Figure 4-2 is set to Read, when it executes it outputs a Boolean True if the control is visible or a Boolean False if it is invisible.

**Tip** Some properties are read-only (such as the Label property) or write only, such as the Value (Signaling) property.

To add terminals to the node, right-click and select **Add Element** from the shortcut menu or use the Positioning tool to resize the node. Then, you can associate each Property Node terminal with a different property from its shortcut menu.

**Tip** Property Nodes execute each terminal in order from top to bottom.

Some properties use clusters. These clusters contain several properties that you can access using the cluster functions. Writing to these properties as a group requires the Bundle function and reading from these properties requires the Unbundle function. To access bundled properties, select **All Elements** from the shortcut menu. For example, you can access all the elements in the Position property by selecting **Properties»Position»All Elements** from the shortcut menu.

However, you also can access the elements of the cluster as individual properties, as shown in Figure 4-3.
Exercise 4-1  Temperature Limit VI

Goal

Use Property Nodes to change the properties of front panel objects programmatically.

Scenario

Complete a VI that records temperature to a waveform chart. During execution, the VI performs the following tasks:

- Set the Δx value of the chart to the user-defined value.
- Clear the waveform chart so it initially contains no data.
- Change the color of a plot if the data exceeds a certain value.
- Make an alarm indicator blink if the data exceeds a certain value.

Design

This VI is already built. You add the following Property Nodes:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform Chart</td>
<td>Temperature</td>
<td>XScale.Multiplier</td>
</tr>
<tr>
<td>Waveform Chart</td>
<td>Temperature</td>
<td>History</td>
</tr>
<tr>
<td>Waveform Chart</td>
<td>Temperature</td>
<td>Active Plot 0»Plot.Color</td>
</tr>
<tr>
<td>Boolean Indicator (LED)</td>
<td>Over Limit</td>
<td>Blinking</td>
</tr>
</tbody>
</table>
Implementation

1. Open Temperature Limit.vi located in the C:\Exercises\LabVIEW Basics II\Temperature Limit directory. The front panel is already built for you.

![Image of the Temperature Limit Front Panel](image1.png)

**Figure 4-4.** Temperature Limit Front Panel

2. Open the block diagram of the VI. A portion has been built for you. Figure 4-5 shows an example of the final block diagram.

![Image of the Temperature Limit Block Diagram](image2.png)

**Figure 4-5.** Temperature Limit Block Diagram
3. Modify the VI so that it sets the $\Delta x$ value of the chart to the $\Delta t$ (ms) value input by the user.

- Right-click the Temperature terminal and select **Create» Property Node»X Scale»Offset and Multiplier»Multiplier** from the shortcut menu to create a Property Node.

- Place the new Property Node to the left of the While Loop.

- Right-click the Property Node and select **Change All To Write** from the shortcut menu.

- Divide $\text{delta t (ms)}$ by 1000 to determine the X-Scale Multiplier, as shown in Figure 4-5.

4. Modify the VI to clear old data from the Temperature chart before starting the temperature acquisition.

   **Tip**  
   To clear a waveform chart from the block diagram, send an empty array of data to the History Data property.

- Resize the Property Node to two terminals.

- Select the **History Data** property in the second terminal.

- Verify that the History Data property is set to Write.

- Right-click the History Data property and select **Create»Constant** from the shortcut menu.

- Wire the Property Node as shown in Figure 4-5.

5. Modify the VI so that when the VI acquires data, it turns the Data trace red and the Over Limit LED blinks when the temperature exceeds the limit value.

- Right-click the Temperature terminal and select **Create» Property Node»Active Plot** from the shortcut menu to create another Property Node.

- Place the new Property Node in the True case of the Case structure.

- Resize the node to two terminals.

- Click the second terminal and select **Plot»Plot Color**.

- Right-click the Property Node and select **Change All To Write** from the shortcut menu.
Wire a numeric constant with a value of 0 to the Active Plot property to select the first plot on the Temperature chart.

Wire the Red Color Box constant to the Plot Color property to set the plot color to red when the data rises above the High Limit.

Create a copy of the Property Node by pressing <Ctrl> while selecting and dragging the Property Node.

Tip  Do not use the clipboard (Edit»Copy) to create a copy of the Property Node. This creates a different type of Property Node that you learn about in the Control References section.

Place the copy of the Property Node in the False case of the Case structure, as shown in Figure 4-6.

Wire a numeric constant with a value of 0 to the Active Plot property to select the first plot on the Temperature chart.

Connect the Green Color Box constant to the Plot Color property to set the plot color to green when the data is below the High Limit.

Figure 4-6.  False Case in the Temperature Limit VI

6. Modify the VI so that when the VI acquires data, the Over Limit LED blinks when the temperature exceeds the limit value.

Right-click the Over Limit terminal and select Create»Property Node»Blinking from the shortcut menu.

Place the new Property Node inside the While Loop.

Right-click the Property Node and select Change All To Write from the shortcut menu.

Wire the Property Node as shown in Figure 4-5.

7. Save the VI.
Lesson 4  Controlling the User Interface

Testing

1. Run the VI to confirm that it behaves correctly.

2. Close the VI.

End of Exercise 4-1
C. Control References

A Property Node created from the front panel object or block diagram terminal is an implicitly-linked Property Node. This means that the Property Node is linked to the front panel object. What if you must place your Property Nodes in a subVI? Then the objects are no longer located on the front panel of the VI that contains the Property Nodes. In this case, you need an explicitly-linked Property Node. You create an explicitly-linked Property Node by wiring a reference to a generic Property Node.

If you are building a VI that contains several Property Nodes or if you are accessing the same property for several different controls and indicators, you can place the Property Node in a subVI and use control references to access that node. A control reference is a reference to a specific front panel object.

This lesson shows one way to use control references. Refer to the Controlling Front Panel Objects topic of the LabVIEW Help for more information about control references.

Creating a SubVI with Property Nodes

As shown in Figure 4-7, the simplest way to create explicitly-linked Property Nodes is to complete the following steps:

1. Create your VI.
2. Select the portion of the block diagram that is in the subVI, as shown in the first part of Figure 4-7.
3. Select Edit»Create SubVI. LabVIEW automatically creates the control references needed for the subVI.
4. Customize and save the subVI. As you can see in the second part of Figure 4-7, the subVI uses the default icon and connector pane.

Figure 4-7. Using Edit»Create SubVI to Create Control References
Figure 4-8 shows the subVI created. Notice that the front panel Control Refnum controls have been created and connected to a Property Node on the block diagram.

![Created SubVI Front Panel](image1)

![Created SubVI Block Diagram](image2)

**Figure 4-8.** Sub VI Created Using Edit-Create SubVI

**Note** A red star on the Control Reference control indicates that the refnum is strictly typed. Refer to the *Strictly Typed and Weakly Typed Control Refnums* section of the *Controlling Front Panel Objects* topic of the *LabVIEW Help* for more information about weakly and strictly typed control references.

**Creating Control References**

To create a control reference for a front panel object, right-click the object or its block diagram terminal and select *Create>Reference* from the shortcut menu.

You can wire this control reference to a generic Property Node. You can pass the control reference to a subVI using a control refnum terminal.
Using Control References

Setting properties with the control reference method is useful for setting the same property for multiple controls. Some properties apply to all classes of controls, such as the Disabled property. Some properties are only applicable to certain control classes, such as the Lock Boolean Text in Center property.

The following example shows how to construct a VI that uses a control reference on the subVI to set the Enable/Disable state of a control on the main VI front panel.

![Diagram showing control reference](image)

Figure 4-9. Control References

The main VI sends a reference for the digital numeric control to the subVI along with a value of zero, one, or two from the enumerated control. The subVI receives the reference by means of the **Ctl Refnum** on its front panel. Then, the reference is passed to the Property Node. Because the Property Node now links to the numeric control in the main VI, the Property Node can change properties of that control. In this case, the Property Node manipulates the Enabled/Disabled state.

Notice the appearance of the Property Node in the block diagram. You cannot select a property in a generic Property Node until the class is chosen. The class is chosen by wiring a reference to the Property Node. This is an example of an explicitly-linked Property Node. It is not linked to a control until the VI is running and a reference is passed to the Property Node. The advantage of this type of Property Node is its generic nature. Because it has no explicit link to any one control, it may be reused for many different controls. This generic Property Node is available on the **Functions** palette.
Selecting the Control Type

When you add a Control Refnum to the front panel of a subVI, you next need to specify the VI Server Class of the control. This specifies the type of control references that the subVI will accept. In the previous example, Control was selected as the VI Server Class type, as shown in Figure 4-9. This allows the VI to accept a reference to any type of front panel control.

However, you can specify a more specific class for the refnum to make the subVI more restrictive. For example, you can select Digital as the class, and the subVI only can accept references to numeric controls of the class Digital. Selecting a more generic class for a control refnum allows it to accept a wider range of objects, but limits the available properties to ones that apply to all objects which the Property Node can accept.

To select a specific control class, right-click the control and select Select VI Server Class ➤ Generic ➤ GObject ➤ Control from the shortcut menu. Then, select the specific control class.
Exercise 4-2  Set Plot Names

Goal

Use control references to create a subVI that modifies graph or chart properties.

Scenario

Create a subVI that allows you to assign a list of plot names to a chart or graph. The subVI should resize the plot legend as necessary to display all of the plots.

Design

Inputs and Outputs

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Reference to a GraphChart object.</td>
<td>Graph Reference</td>
<td>N/A</td>
</tr>
<tr>
<td>1-D Array of Strings Control</td>
<td>Plot Names</td>
<td>Empty Array</td>
</tr>
<tr>
<td>Error Cluster Control</td>
<td>Error In</td>
<td>No Error</td>
</tr>
<tr>
<td>Error Cluster Indicator</td>
<td>Error Out</td>
<td>No Error</td>
</tr>
</tbody>
</table>

Control References

The only class that contains both the Waveform Chart and the Waveform Graph is the GraphChart class. In order to write a subVI that can accept references to both charts and graphs you must use a weakly typed control reference of the GraphChart class. However, this class also contains other charts and graphs, such as the XY Graph. This subVI, generates an error if the user wires any type of graph other than a Waveform Chart or a Waveform Graph. You can determine if the user has wired the correct type by using the ClassName property to control a Case structure. If the correct class is wired, use the To More Specific Class function to get a reference to the appropriate subclass. After you have a reference to a WaveformChart or a WaveformGraph you can set the properties to modify plot names.
Properties
Graphs and charts do not have a single property to set all of the plot names. Instead you must use a combination of properties to set each plot name. In this exercise, use the following properties and methods:

**ClassName**—This property returns a string indicating the control class of the object that the property is called on. You can access this property for any control.

**LegAutosize**—This property controls whether the Graph Legend automatically resizes to accommodate the plot names within it. Before modifying the plot names you should set this property to False. Otherwise, the legend may resize in such a way that it is separated from the graph or covers the graph or other controls.

**LegPlots**—This property controls the number of plots visible on the Graph Legend. When adding your legend to the front panel, remember to leave room for the legend to expand when you set this property. The legend expands downwards.

**ActPlot**—Properties affecting a plot act upon one plot at a time. This property controls the active plot. Any time a plot property is set or read it applies to the active plot. The plots are numbered sequentially as they are created, starting with zero.

**Plot.Name**—This property sets the name of the active plot.

Implementation
1. Open a blank VI.
2. Save the VI as *Set Plot Names.vi* in the `C:\Exercises\LabVIEW Basics II\Set Plot Names` directory.
3. Create the front panel window.
   - Add a Control Refnum to the front panel window.
   - Name the Control Refnum `Graph Reference`.
   - Right-click `Graph Reference` and choose `Select VI Server Class»Generic»GObject»Control»GraphChart»GraphChart` from the shortcut menu.
   - Add an Array to the front panel window.
   - Name the array `Plot Names`. 
❑ Add a String Control to the **Plot Names** array.

❑ Add an Error In cluster.

❑ Add an Error Out cluster.

❑ Arrange the controls as shown in Figure 4-10.

![Figure 4-10. Set Plot Names Front Panel](image)

**Tip** Because the front panel of this subVI is not displayed to the user, you do not have to put as much effort into making it visually appealing. You should always organize your front panels logically. However, you should not spend too much time on panels that the user does not see.

4. Switch to the block diagram.

5. Identify the class of the control reference and generate an error if it has an invalid class.

![Figure 4-11. Default Case](image)
6. On the **Functions** palette, select the **Programming»Application Control** category. Most of the functions you use in this section come from this palette.

- Add a Property Node to the block diagram.
- Wire **Graph Reference** to the **reference** input of the Property Node.
- Select **Class Name** in the **property** section of the Property Node.
- Add a Case structure to the block diagram as shown in Figure 4-11.
- Wire the **ClassName** output of the Property Node to the case selector of the Case structure.
- Switch to the False case of the Case structure.
- Delete the **False** text in the case name so that the case name resembles Figure 4-11.

**Note** The Default case of the Case structure is selected if the class of the control reference does not match one of the other cases. In this case, if the default case executes, then the control reference passed to this subVI is not a WaveformGraph or a WaveformChart. Remember for a Case structure, the case selector label is case sensitive.

- Add an Error Cluster From Error Code VI to the Case structure.
- Right-click the **error code** input of the Error Cluster From Error Code VI and select **Create»Constant** from the shortcut menu.
- Enter 1057 in the constant.

**Note** Error code 1057 corresponds to the message **Object cannot be typecasted to the specified type**. This is the appropriate error to generate if the caller of the subVI passes a control reference of the wrong class.

- Wire the diagram as shown in Figure 4-11.
7. Handle the WaveformGraph references.

![WaveformGraph Case](image)

**Figure 4-12.** WaveformGraph Case

- Switch to the True case of the Case structure.
- Change the True text in the case name to WaveformGraph so that the case name resembles Figure 4-12.

⚠️ **Caution** The text entered the case selector label must *exactly* match the input string, including spaces and case. For this example, enter WaveformGraph.

- Add a To More Specific Class function to the Case structure.
- Right-click the **target class** input of the To More Specific Class function and select Create>Constant from the shortcut menu.
- Click the constant you created in the previous step and select Generic>GObject>Control>GraphChart>WaveformGraph>WaveformGraph class.
- Add a Property Node to the Case structure.
- Wire the specific class reference output of the To More Specific Class function to the reference input of the Property Node.
- Click the **Property** section of the Property Node and select Legend>Autosize.
- Expand the Property Node to display two properties.
- Click the second property in the Property Node and select Legend>Plots Shown.
❑ Right-click the Property Node and select **Change All To Write** from the shortcut menu.

❑ Right-click the **LegAutosize** property and select **Create»Constant** from the shortcut menu. Ensure that the value of the constant is False.

❑ Add an Array Size function to the Case structure.

❑ Add a For Loop to the Case structure.

❑ Add a Property Node to the For Loop.

❑ Wire the **dup reference** output of the first Property Node through the border of the For Loop to the **reference input** of the second Property Node.

❑ Click the **Property** section of the Property Node and select **Active Plot** from the list.

❑ Expand the Property Node to display two properties.

❑ Click the second property in the Property Node and select **Plot» Plot Name** from the list.

❑ Right-click the Property Node and select **Change All To Write** from the shortcut menu.

❑ Wire the diagram as shown in Figure 4-12. Use shift registers when wiring the error wires through the For Loop and use Auto Indexing for the Plot Names wire.
8. Handle WaveformChart references.

![Diagram of WaveformChart Case]

**Figure 4-13. WaveformChart Case**

- Right-click the border of the Case structure and select **Duplicate Case** from the shortcut menu.

- Enter **WaveformChart** in the case name.

  **Caution** The text entered the case selector label must exactly match the input string, including spaces and case. For this example, enter **WaveformChart**.

- Click the WaveformGraph reference constant and select **Generic» GObject»Control»GraphChart»WaveformChart**.

  **Note** When you change the class of a control reference, all Property Nodes and Invoke Nodes using the reference become invalid because the properties refer to a class that does not match the reference. Notice that all of the property names change to black when you change the class reference and that the run arrow is broken. Leave the broken wires alone, because the wires reconnect as you reselect the properties.

- Click each of the four properties and select the correct property again. The four properties are **Legend»Autosize**, **Legend»Plots Shown**, **Active Plot**, and **Plot»Plot Name**. The resulting diagram appears as shown in Figure 4-13.
9. Create the icon and connector pane for the subVI. Figure 4-14 shows an example icon and connector pane.

![Set Plot Names VI](image)

Table: Connections vs. Connector Pane

<table>
<thead>
<tr>
<th>Connections</th>
<th>Connector Pane</th>
</tr>
</thead>
<tbody>
<tr>
<td>❑ Switch to the front panel of the VI.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Right-click the VI Icon and select Show Connector from the shortcut menu.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Right-click the connector pane and select Patterns from the shortcut menu to choose a pattern.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Wire the connector pane.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Right-click the connector pane and select Show Icon from the shortcut menu.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Right-click the icon and select Edit Icon from the shortcut menu.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Use the tools in the Icon Editor to create an icon.</td>
<td>❑</td>
</tr>
<tr>
<td>If you prefer to use a pre-built icon, select Edit Import Picture From File and navigate to C:\Exercises\LabVIEW Basics II\Set Plot Names\Set Plot Names Icon.bmp. Select Edit Paste.</td>
<td>❑</td>
</tr>
<tr>
<td>❑ Close the Icon Editor when you are finished.</td>
<td>❑</td>
</tr>
</tbody>
</table>

10. Save the VI.
Testing

1. Test the VI using a Waveform Graph.

- Create a blank VI.
- Add a Waveform Graph to the front panel window.
- Open the block diagram.
- Right-click the Waveform Graph terminal and select Create»Reference from the shortcut menu.
- Add the Set Plot Names VI to the block diagram of the new VI.
- Wire the WaveformGraph reference to the Graph Reference input terminal of the Set Plot Names VI.
- Right-click the Plot Names input of the Set Plot Names VI and select Create»Control from the shortcut menu. The block diagram should look something like Figure 4-15.
- Switch to the front panel window of the new VI.
- Enter One and Two as items in the Plot Names array.
- Move the Plot Legend to the right of the graph so that you can expand the legend.
- Run the VI. One and Two appear in the legend.
2. Test the VI using a Waveform Chart.

- Right-click the Waveform Graph and select Replace»
  Graph Indicators»Waveform Chart from the shortcut menu.

- Add Three as another item in the Plot Names array.

- Run the VI. Three appears in the legend of the chart.

3. Test the VI with a XY Graph.

- Right-click the Waveform Graph and select Replace»
  Graph Indicators»XY Graph from the shortcut menu.

- Add Four as another item in the Plot Names array.

- Run the VI. A typecasting error occurs.

4. Close the VI. You do not need to save the VI used for testing the Set Plot Names VI.

End of Exercise 4-2
D. Invoke Nodes

Invoke Nodes access the methods of an object.

Use the Invoke Node to perform actions, or methods, on an application or VI. Unlike the Property Node, a single Invoke Node executes only a single method on an application or VI. Select a method by using the Operating tool to click the method terminal or by right-clicking the white area of the node and selecting **Methods** from the shortcut menu. You also can create an Invoke Node by right-clicking the object, selecting **Create»Invoke Node**, and selecting a method from the shortcut menu.

The name of the method is always the first terminal in the list of parameters in the Invoke Node. If the method returns a value, the method terminal displays the return value. Otherwise, the method terminal has no value.

The Invoke Node lists the parameters from top to bottom with the name of the method at the top and the optional parameters, which are dimmed, at the bottom.

**Example Methods**

An example of a method common to all controls is the Reinitialize to Default method. Use this method to reinitialize a control to its default value at some point in your VI. The VI class has a similar method called Reinitialize All to Default.

Figure 4-16 is an example of a method associated with the Waveform Graph class. This method exports the waveform graph image to the clipboard or to a file.

![Invoke Node for the Export Image Method](image)
Exercise 4-3  Front Panel Properties VI

Goal

Affect the attributes of a VI by using Property Nodes and Invoke Nodes.

Scenario

You can set the appearance properties of a VI statically by using the VI properties page. However, robust user interfaces often must modify the appearance of a front panel while the program runs.

You must create a VI that can perform the following tasks on demand:

- Show or hide its title bar
- Show or hide its menu bar
- Become transparent so that objects behind the VI can be seen
- Move to the center of the screen

Design

Inputs and Outputs

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Toggle Switch</td>
<td>Show Menu Bar?</td>
<td>True (Yes)</td>
</tr>
<tr>
<td>Vertical Toggle Switch</td>
<td>Show Title Bar?</td>
<td>True (Yes)</td>
</tr>
<tr>
<td>Vertical Toggle Switch</td>
<td>Make VI Transparent?</td>
<td>True (Yes)</td>
</tr>
<tr>
<td>OK Button</td>
<td>Center</td>
<td>False</td>
</tr>
<tr>
<td>Stop Button</td>
<td>Stop</td>
<td>False</td>
</tr>
</tbody>
</table>

Tip  Use the vertical toggle switches because their default mechanical action is switch when pressed. Use the OK button because its default action is latch when released.

Properties

Use the following properties and methods on the VI class:

ShowMenuBar—When this property is true, the menu bar of the VI is visible.

Figure 4-17.  VI Menu Bar
TitleBarVisible—When this property is true, the title bar of the VI is visible.

![Front Panel Properties.vi Front Panel](image)

**Figure 4-18.** VI Title Bar

RunVITransparently—When this property is true, the transparency of the VI can vary. The default value of this property is FALSE, so you must write a TRUE to this property before varying the transparency of the VI.

Transparency—This property varies the transparency of the VI. The property accepts any value between 0 and 100. A value of 0 makes the VI completely opaque (normal behavior), and a value of 100 makes the VI completely transparent (invisible). For this exercise, you set the value to 50 when the **Make VI Transparent?** button is clicked.

Methods

Unlike properties, a method has an effect every time you call it. Therefore, you should only call methods when you want to perform an action. For example, if you call the Fp.Center method during each iteration of a loop, the VI is continually centered, thereby preventing the user from moving it. You can use a Case structure to control calling the method in a given iteration of a loop. Use the following method on the VI class:

Center—Each time this method is called, the VI moves to the center of the screen.

Tip Use the **Context Help** window to view descriptions of each property and method.

VI Structure

The VI polls the front panel controls every 50 milliseconds and sets the value of the properties based on the current value of the controls. A Case structure controls the execution of the Center method.
Implementation

In the following steps, you create the front panel window for the VI. An example of the front panel window is shown in Figure 4-19.

![Front Panel Properties VI Front Panel Window](image)

Figure 4-19. Front Panel Properties VI Front Panel Window

1. Open a blank VI.

2. Save the VI as Front Panel Properties.VI in the C:\Exercises\LabVIEW Basics II\Front Panel Properties directory.

   - Add a Vertical Toggle Switch to the front panel window.
   - Name the switch Show Menu Bar?.
   - Create free labels for the Yes and No states of the switch.

4. Create the Show Title Bar? switch.
   - Make a copy of the Show Menu Bar? switch.
   - Rename the switch Show Title Bar?.
   - Copy the free labels for the Yes and No states from the Show Menu Bar? switch.

5. Create the Make VI Transparent? switch.
   - Make a copy of the Show Menu Bar? switch.
   - Rename the switch Make VI Transparent?.
   - Copy the free labels for the Yes and No states from the Show Menu Bar? switch.
6. Create the **Center** button.
   - Add an OK button to the front panel window.
   - Name the button **Center**.
   - Change the Boolean text on the button to **Center the VI**.
   - Right-click the button and select **Visible Items»Label** from the shortcut menu to hide the label.

7. Create the **Stop** button.
   - Add a Stop button to the front panel window.
   - Right-click the button and select **Visible Items»Label** from the shortcut menu to hide the label.

8. Select **Edit»Make Current Values Default**.


In the following steps, create the block diagram for the VI. An example of the block diagram is shown in Figure 4-20.

![Figure 4-20. Front Panel Properties Block Diagram](image)

![Figure 4-21. False Case for Center Method](image)
10. Add a While Loop from the **Structures** category around the terminals.

11. Create a reference to the VI.

   - Add a VI Server Reference to the block diagram to the left of the While Loop.
   - Set the VI Server Reference to **This VI** if it is not already.

   **Note**  The This VI reference allows you to access all the methods and properties of the current VI without having to explicitly open and close a reference.

12. Create a Property Node for the RunTransparently property.

   - Right-click the **This VI** reference and select **Create»Property»Front Panel Window»Run VI Transparently** from the shortcut menu to create a Property Node.
   - Move the Property Node to the right of the This VI reference, outside of the While Loop.
   - Right-click the Property Node and select **Change All to Write** from the shortcut menu.
   - Right-click the **FP.RunTransparently** property and select **Create»Constant** from the shortcut menu.
   - Change the value of the constant to True.

13. Create a Property Node for the ShowMenuBar, TitleBarVis, and Transparency properties.

   - Right-click the This VI reference and select **Create»Property»Front Panel Window»Show Menu Bar** from the shortcut menu to create another Property Node.
   - Expand the Property Node to show three elements.
   - Click the second item in the Property Node and select **Front Panel Window»Title Bar Visible**.
   - Click the third item in the Property Node and select **Front Panel Window»Transparency**.
   - Right-click the Property Node and select **Change All to Write** from the shortcut menu.
   - Move the Property Node inside the While Loop.
Place a Select function inside the While Loop.

Place two Numeric Constants with values 0 and 50 to the left of the Select function.

Wire the 0 numeric constant to the f terminal of the Select function.

Wire the 50 numeric constant to the t terminal of the Select function.

Wire the Boolean controls to the appropriate properties, as shown in Figure 4-20.

14. Create a Invoke Node for the Center method.

Right-click the This VI reference and select Create » Method » Front Panel » Center from the shortcut menu to create an Invoke Node.

15. Add a Case structure around the FP.Center Invoke Node.

16. Add a 50 ms wait to the loop.

Add a Wait (ms) function in the While Loop.

Right-click the milliseconds to wait input and select Create » Constant from the shortcut menu.

Enter 50 in the constant.

17. Set the While Loop to stop when the user clicks the Stop button or when an error occurs.

Add an Unbundle By Name function in the While Loop.

Add an Or function in the While Loop.

18. Wire the diagram as shown in Figure 4-20 and Figure 4-21. Make sure to replace the error cluster tunnel with a shift register.

19. Display any errors that may occur to the user.

Add a Simple Error Handler VI to the right of the While Loop.

Wire the Simple Error Handler VI to the error cluster output shift register from the While Loop.

20. Save the VI.
Testing

1. Switch to the front panel window of the VI.
2. Run the VI.
3. Try each of the buttons and observe the results.

End of Exercise 4-3
Self-Review: Quiz

1. For each of the following items, determine whether they operate on a VI class or a Control class.
   - Format and Precision
   - Blinking
   - Reinitialize to Default Value
   - Show Tool Bar

2. You have a ChartGraph control refnum, shown at left, in a subVI. Which of the following control references could you wire to the control refnum terminal of the subVI? (multiple answers)
   a. Control reference of an XY Graph
   b. Control reference of a Numeric Array
   c. Control reference of a Waveform Chart
   d. Control reference of a Boolean Control
Self-Review: Quiz Answers

1. For each of the following items, determine whether they operate on a VI class or a Control class.
   - Format and Precision: **Control**
   - Blinking: **Control**
   - Reinitialize to Default Value: **Control**
   - Show Tool Bar: **VI**

2. You have a ChartGraph control refnum, shown at left, in a subVI. Which control references could you wire to the control refnum terminal of the subVI?
   a. **Control reference of an XY Graph**
   b. Control reference of a Numeric Array
   c. **Control reference of a Waveform Chart**
   d. Control reference of a Boolean Control