LabVIEW™ Upgrade Notes

Version 7.0

These upgrade notes describe the process of upgrading LabVIEW for Windows, Mac OS, and UNIX to version 7.0, issues you might encounter when you upgrade, and new features, including several enhancements to the LabVIEW environment.

Refer to the *LabVIEW Environment Enhancements* section of this document for more information about each enhancement to the LabVIEW environment and for information about customizing the LabVIEW environment. National Instruments recommends that all upgrade users read the *LabVIEW Environment Enhancements* section before using LabVIEW 7.0. Refer to the *Getting Started with LabVIEW* manual for exercises you can complete to familiarize yourself with the new features and enhancements to the LabVIEW environment in LabVIEW 7.0.

For more information…

Refer to the *LabVIEW User Manual* and the *LabVIEW Help* for more information about LabVIEW 7.0 features. Access the *LabVIEW Help* by selecting **Help»VI, Function, and How-To-Help**. Access a PDF version of the *LabVIEW User Manual* and all other LabVIEW manuals by selecting **Help»Search the LabVIEW Bookshelf**. Use the *LabVIEW Bookshelf* to search PDF versions of all the LabVIEW manuals and Application Notes.

You must have Adobe Acrobat Reader with Search and Accessibility 5.0.5 or later installed to view the PDFs. Refer to the Adobe Systems Incorporated Web site at `www.adobe.com` to download Acrobat Reader.

Contents

April 2003
321780E-01

NATIONAL INSTRUMENTS™

Upgrade Issues

If you are upgrading from LabVIEW 6.*x*, refer to the *Converting VIs*, the *Upgrading Toolsets, Instrument Drivers, and Add-Ons*, and the *Upgrading from LabVIEW 6.x* sections of this document first.

If you are upgrading from LabVIEW 5.*x*, refer to the *Converting VIs*, the *Upgrading Toolsets, Instrument Drivers, and Add-Ons*, and the *Upgrading from LabVIEW 5.x* sections of this document first.

If you are upgrading from LabVIEW 4.*x*, refer to the *Converting VIs*, the *Upgrading Toolsets, Instrument Drivers, and Add-Ons*, and the *Upgrading from LabVIEW 4.x* sections of this document first.

If you are upgrading from LabVIEW 3.*x*, refer to *Upgrading Toolsets, Instrument Drivers, and Add-Ons* and the *Upgrading from LabVIEW 3.x or Earlier Versions* sections of this document first.

Converting VIs

When you open a VI last saved in LabVIEW 4.0 or later, LabVIEW 7.0 automatically converts and compiles the VI. You must save the VI in LabVIEW 7.0, or the conversion process, which uses extra memory resources, occurs every time you access the VI.

Note VIs you save in LabVIEW 7.0 do not load in earlier versions of LabVIEW. Select **File»Save with Options** and select the **Save for Previous** option to save VIs so they can run in LabVIEW 6.1. Before saving VIs in LabVIEW 7.0, keep a backup copy of VIs you plan to use in LabVIEW 6.0 or earlier.

You can estimate the amount of memory required to convert VIs by totalling the amount of memory that the VIs and all their subVIs occupy on disk. If the VIs are in VI libraries, add approximately 30 percent of the VI library size because the VIs are compressed. The conversion process might require at least that much memory and an additional 3 MB of memory to run LabVIEW.

If your computer does not have enough memory to convert all the VIs at once, convert the VIs in stages. Examine the hierarchy of VIs you want to convert and begin by loading and saving subVIs in the lower levels of the hierarchy. Then progress gradually to the higher levels of the hierarchy. You also can select **Tools»Advanced»Mass Compile** to convert a directory of VIs. However, mass compiling converts VIs in a directory or VI library in alphabetical order. If the conversion process encounters a high-level VI first, mass compiling requires approximately the same amount of memory as if you opened the high-level VI first.

You can monitor memory usage by selecting **Help»About LabVIEW** to display a summary of the amount of memory you have used.

Upgrading Toolsets, Instrument Drivers, and Add-Ons

After you install LabVIEW 7.0, make sure you have the latest version of any add-ons and reinstall the add-ons in the LabVIEW 7.0 directory.

You also must mass compile existing toolset VIs for use in LabVIEW 7.0. Refer to the *Converting VIs* section of this document for more information about mass compiling VIs. LabVIEW 7.0 is compatible with toolsets, instrument drivers, and add-ons designed for LabVIEW 4.0 and later, with the following exceptions:

- **(Full Development System) LabVIEW Application Builder**—You must upgrade to LabVIEW Application Builder 7.0. The LabVIEW Professional Development System version 7.0 includes Application Builder 7.0 libraries. Refer to the *LabVIEW Application Builder User*

Guide for more information about installing the LabVIEW Application Builder.

- **(Full Development System) LabVIEW Professional G Developers Toolkit**—If you have the Professional G Developers Toolkit 5.0 or later, you must upgrade to the LabVIEW Professional Development System version 7.0. This upgrade is free to existing users of the Professional G Developers Toolkit 5.1. The Professional Development System version 7.0 includes the features of the Professional G Developers Toolkit.

Upgrading Previous Versions of LabVIEW

Upgrading to new versions of LabVIEW does not affect previous versions of LabVIEW on the computer because the new versions install in a different directory. LabVIEW 5.*x* and earlier installs in the `labview` directory. LabVIEW 6.0 and later installs in the `labview` *`x.x`* directory, where *`x.x`* is the version number.

To use LabVIEW environment settings from a previous version of LabVIEW, copy the LabVIEW preferences file from the `labview` directory in which the previous version is installed. After you install LabVIEW 7.0, copy the LabVIEW preferences file from the previous version into the LabVIEW 7.0 directory.

(Windows) LabVIEW stores preferences in the `labview.ini` file.

(Mac OS) LabVIEW stores preferences in the `LabVIEW Preferences` text file in the `System:Preferences` folder.

(UNIX) LabVIEW stores preferences in the `.labviewrc` file in your home directory.

To use files from the `user.lib` directory of a previous version of LabVIEW, copy the contents from the `labview` directory in which the previous version is installed. After you install LabVIEW 7.0, copy the files to the `user.lib` directory in the LabVIEW 7.0 directory.

(Windows) You also can replace the existing version of LabVIEW with LabVIEW 7.0 by using the Add/Remove Programs applet in the Control Panel to uninstall the existing version of LabVIEW. The uninstaller does not remove any files you created in the `labview` directory.

Caution If you saved your own VIs and controls in existing `.llb` files in the `vi.lib` directory, LabVIEW uninstalls the `.llb` files including any VIs and controls you saved in the `.llb` files.

Run the LabVIEW 7.0 installer and set the default installation directory to the same `labview` directory where you installed the previous version of LabVIEW.

Refer to the following sections for upgrade and compatibility issues specific to different versions of LabVIEW.

Upgrading from LabVIEW 6.*x*

You might encounter the following issues when you upgrade to LabVIEW 7.0 from LabVIEW 6.*x*.

Windows 95 Support

LabVIEW 7.0 does not support Windows 95.

HiQ Support

National Instruments does not support HiQ functionality on Mac OS in LabVIEW 7.0 and will no longer support HiQ functionality on all platforms after LabVIEW 7.0. If an application uses HiQ VIs, consider replacing them with the Analyze and Mathematics VIs. Refer to the *LabVIEW Help* for information about using the Analyze and Mathematics VIs.

Serial Compatibility VIs

The Serial Compatibility VIs do not appear on the **Functions** palette. Use the VISA VIs and functions to build VIs that communicate with VXI devices.

LabVIEW no longer uses the `serpdrv` driver to communicate with the serial driver of the operating system. LabVIEW includes compatible VIs based on VISA. For new applications, use the VISA and Serial VIs and functions to control serial devices. Any VIs built in previous versions of LabVIEW that include Serial VIs continue to work in LabVIEW 7.0.

If you reconfigured the mapping of port numbers to ports, you must specify a mapping to those ports. Use the set serial alias ports VI in the `labview\vi.lib\Instr\_sersup.llb` to specify the serial port mappings. Wire a string array to the **VISA Aliases** input of the VI and enter the port names you use in the input array. Each element in the array should correspond to a port. For example, if you configured port 0 to map to the VISA alias MySerialPort, enter `MySerialPort` as the first element of the **VISA Aliases** input array. You must call the Set Serial Alias Ports VI before you call the VISA Configure Serial Port VI.

Refer to the `examples\instr\smplserl.llb` for examples of using the VISA VIs and functions to control serial instruments.

Default Data in Loops

In LabVIEW 6.0 and earlier, For Loops produced undefined data if the loop did not execute. In LabVIEW 6.1 and later, For Loops produce default data if you wire `0` to the count terminal of the For Loop or if you wire an empty array to the For Loop as an input with auto-indexing enabled. The loop does not execute, and any output tunnel with auto-indexing disabled contains the default value for the tunnel data type.

Remote Front Panel License

The LabVIEW Full Development System and the Application Builder include a remote front panel license that allows one client to view and control a front panel remotely. The LabVIEW Professional Development System includes a remote front panel license that allows five clients to view and control a front panel remotely.

(Windows 98) You cannot upgrade the remote front panel license to support more clients. **(Windows 2000/NT/XP/Me, Mac OS, and UNIX)** You can upgrade the remote front panel license to support more clients.

Multiple Thread Allocation

LabVIEW 7.0 allocates more threads for executing VIs than in earlier versions of LabVIEW. Because of this change, you might encounter errors with multiple threads if you incorrectly mark Call Library Function Nodes as reentrant when the DLL you call is not actually reentrant. Refer to Chapter 2, *Shared Libraries (DLLs)*, of the *Using External Code in LabVIEW* manual for more information about reentrancy.

To change how LabVIEW allocates threads, use the Thread Configuration VI located in the `vi.lib\Utility\sysinfo.llb`. You also can disable reentrancy for VIs by removing the checkmark from the **Reentrant execution** checkbox in the **Execution** page of the **VI Properties** dialog box.

Instrument Drivers

The LabVIEW package no longer includes the LabVIEW Instrument Driver Library CD, which contains instrument drivers. Download instrument drivers from the National Instruments Instrument Driver Network at `ni.com/idnet`. The National Instruments Device Drivers CD includes NI-DAQ, NI-VISA, and other National Instruments drivers.

Units and Conversion Factors

After using the Compound Arithmetic function, you no longer need to use the Convert Unit function to remove the extra unit.

The unit conversion factors in LabVIEW 7.0 more closely match the guidelines published by the National Institute for Standards and Technology (NIST) in the *Guide for the Use of the International System of Units (SI)*. Also, the `calorie` unit is now `calorie (thermal)`, and `horse power` is now `horse power (electric)`. The abbreviations for these units did not change. The following table details the changes in unit conversion factors between LabVIEW 6.1 and 7.0.

Unit	6.1 Definition	7.0 Definition
astronomical unit (AU)	149,498,845,000 m	149,597,900,000 m
British Thermal Unit (mean)	1055.79 J	1055.87 J
electron volt (eV)	1.602e–19 J	1.60217642e–19 J
foot-candle	10.764 lx	10.7639 lx
horse power vs. horse power (electric)	745.7 W	746 W The new conversion is exact.
imperial gallon	4.54596 l	4.54609 l
light year	9.4605 Pm	9.46073 Pm
pound force	4.448 N	4.448222 N
rod	16.5 ft	5.029210 m
slug	32.174 lb	14.59390 kg
unified atomic mass (u)	1.66057e–27 kg	1.66053873e–27 kg

Defer Panel Updates Property

When you set this property to TRUE, LabVIEW redraws any front panel objects with pending changes then defers all new requests for front panel updates. In LabVIEW 6.1 and earlier, LabVIEW waits until the Defer Panel Updates property is FALSE to redraw any front panel objects with pending changes.

In some cases, this change can cause LabVIEW to redraw the changed elements of the front panel an extra time.

Data Ranges for Numeric Controls

In LabVIEW 6.1 and earlier, some numeric controls were set to a minimum value of `0.00`, a maximum value of `0.00`, an increment value of `0.00`, and an out of range action of **Ignore** as default settings. In LabVIEW 7.0, these

numeric controls are changed to use the default data range values for the data type.

Coercion Dots and Type Definitions

In LabVIEW 6.1 and later, wires include information about type definitions, so you might notice more coercion dots on block diagrams. If you wire a type definition to a VI or function terminal that is not a type definition terminal, a coercion dot appears. A coercion dot also appears if you wire an output terminal that is a type definition to an indicator that is not a type definition. These coercion dots indicate where you are not using type definitions consistently in the VIs.

In this case, coercion dots do not affect run-time performance.

Refer to the *LabVIEW Help* for information about using the Flatten To String function to flatten type definitions.

File Dialog Box Button Label

In LabVIEW 6.1 and earlier, the file dialog box the File Dialog function displays has a button label of **Save** if the user can enter a new filename. Otherwise, the button label is **Open**. In LabVIEW 7.0, the button label is **OK** in all cases unless you change it. Use the **button label** input of the File Dialog function to change the label of the button. If you use the File Dialog function in an existing VI, consider reviewing the behavior of the VI to make sure the default label of **OK** is appropriate to the functionality of the VI.

Control Online Help Function

The **Path to the help file** input of the Control Online Help function is required. You can wire a compiled help filename (`.chm` or `.hlp`) or the full path to a compiled help file to the input. If you wire only a compiled help filename, LabVIEW searches the `labview\help` directory for that file.

Run VI Method

If you set the **Auto Dispose Ref** parameter of the Run VI method to TRUE, LabVIEW disposes the reference even if the method returns an error. This might break a VI if a part of the block diagram depends on the reference.

Displaying the Front Panel When Loaded

In LabVIEW 7.0, if you configure a VI to display its front panel when LabVIEW loads the VI and you load the VI using the VI Server, LabVIEW does not display the front panel. You must use the Open FP method to display the front panel programmatically.

Open VI Reference Function

If you use the Open VI Reference function to create a reference to a template and the template is already in memory, the function returns an error.

IVI Configuration Store File

The IVI Configuration Store file format now requires that all names be case-sensitive. If you use logical names, driver session names, or virtual names in your program, make sure that the name you use matches the name defined in the IVI Configuration Store file exactly, without any variations in the case of the characters in the name.

Technical Support Form

The LabVIEW installer does not install `techsup.llb`. Refer to the National Instruments Web site at `ni.com/support` to solve installation, configuration, and application problems and questions.

Upgrading from LabVIEW 5.*x*

You might encounter the following issues when you upgrade to LabVIEW 7.0 from LabVIEW 5.*x*. Refer to the *Upgrading from LabVIEW 6.x* section of this document for information about other upgrade issues you might encounter.

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between version 5.*x* and 7.0 and to the *LabVIEW 5.1 Addendum* at `ni.com/manuals` for more information about the new features and changes in each version.

Converting Datalog Files

LabVIEW 7.0 checks the type definition of datalog files to determine if a conversion is necessary. If the datalog file is older than LabVIEW 6.0 or contains waveform datatypes, LabVIEW 7.0 converts the file for reading and appending. In all other cases, reading from the datalog file and appending to the datalog file does not convert the datalog file.

When you open a datalog file created in an earlier version of LabVIEW, LabVIEW 7.0 prompts you to convert the file to the LabVIEW 7.0 format. If you choose to convert it, LabVIEW replaces the datalog file with data converted to the new format. If you choose not to convert the file, LabVIEW 7.0 returns an error and does not open the file.

Note You should make a backup copy of datalog files before converting if you plan to continue to use old data in LabVIEW 6.1 or earlier. You cannot revert to or read converted datalog files in LabVIEW 6.1 or earlier.

To automatically convert datalog files when you open them, add the following line to the LabVIEW preferences file:

```
silentDatalogConvert=True
```

(Mac OS) Add the following line:

```
silentDatalogConvert:True
```

(UNIX) Add the following line:

```
labview.silentDatalogConvert:True
```

Set the preference to `False` if you do not want to automatically convert datalog files when you open them.

Compatibility Issues between the LabVIEW 5.*x* VI Server and a LabVIEW 7.0 Client

Attempting to make a connection to the VI Server of a LabVIEW 5.*x* application from a LabVIEW 7.0 client fails because the LabVIEW 5.*x* application does not recognize some new aspects of the LabVIEW 7.0 VI Server protocol.

You can connect to the VI Server of a LabVIEW 7.0 application from a LabVIEW 5.*x* client.

Exponential Representation

In LabVIEW 6.0 and earlier, the ^ operator represented exponentiation in the Formula Node. In LabVIEW 7.0, the operator for exponentiation is `**`—for example, `x**y`. The ^ operator represents the bitwise exclusive or (XOR) operation.

UDP Functions

Use the built-in UDP functions for network communication. The UDP VIs exist as compatibility VIs in the `vi.lib\_oldvers\_oldvers.llb`.

Upgrading from LabVIEW 4.*x*

You might encounter the following issues when you upgrade to LabVIEW 7.0 from LabVIEW 4.*x*. Refer to the *Upgrading from LabVIEW 5.x* and the *Upgrading from LabVIEW 6.x* sections of this document for information about other upgrade issues you might encounter.

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between version 4.*x* and 7.0 and to the *LabVIEW 5.1 Addendum* at `ni.com/manuals` for more information about the new features and changes in each version.

Converting Boolean Data to and from LabVIEW 4.*x*

The format in which LabVIEW stores Boolean data changed between LabVIEW 4.*x* and LabVIEW 5.*x*. LabVIEW 4.*x* stores Boolean data in two bytes unless the data is in an array, in which case LabVIEW 4.*x* stores each Boolean element in a single bit. LabVIEW 7.0 stores a Boolean value in a single byte, regardless of whether it is in an array. This change enables more block diagram functions to support arrays of Boolean values and makes the behavior of these arrays more consistent with the behavior of arrays of numbers. The new Boolean data format affects data manipulation in Code Interface Nodes (CINs), but LabVIEW 7.0 provides compatibility for existing CINs.

If you write binary data that includes one or more Boolean values to a file in LabVIEW 4.*x*, its format is different than if you write the same data in LabVIEW 7.0. LabVIEW 7.0 provides a mechanism for reading binary data written in LabVIEW 4.*x* and writing binary data that LabVIEW 4.*x* can read. The Write File, Read File, Type Cast, Flatten To String, and Unflatten From String functions have a **Convert 4.x Data** shortcut menu item. If you right-click the function and select this menu item, the function treats binary data as if it were written for LabVIEW 4.*x*. To produce data formatted for LabVIEW 4.*x*, use the Write File, Flatten To String, or Type Cast function. To read data formatted for LabVIEW 4.*x*, use the Read File, Unflatten From String, or Type Cast function. When you select the **Convert 4.x Data** shortcut menu item, LabVIEW 7.0 draws a red `4.x` on the function to indicate that it is converting data to or from LabVIEW 4.*x* format. To avoid the conversion of data, select the **Convert 4.x Data** shortcut menu item again to remove the checkmark next to it.

If you have several data files with Boolean values, you can create a VI that opens these files and writes the data to a new data file that LabVIEW 7.0 recognizes.

In LabVIEW 7.0, when you load a VI last saved in LabVIEW 4.*x* or earlier, LabVIEW 7.0 automatically sets the **Convert 4.x Data** attribute on the Write File, Read File, Type Cast, Flatten To String, and Unflatten From String functions. These functions continue to operate as before. When you decide that a VI needs to use the LabVIEW 7.0 Boolean data format, select the **Convert 4.x Data** shortcut menu item on each of these functions. Typically, you should use the LabVIEW 7.0 Boolean data format if VIs do not need to manipulate files that contain Boolean data written in LabVIEW 4.*x* or earlier and do not send or receive data that contain Boolean data to or from VIs running in LabVIEW 4.*x* or earlier. Support for the previous Boolean data format might be discontinued in future versions of LabVIEW.

VI Control VIs

The VI Control VIs do not appear on the **Functions** palette but exist as compatibility VIs in the `vi.lib\utility\victl.llb`. Use the VI Server functions Open VI Reference, Call By Reference, Property Node, and Invoke Node instead of the VI Control VIs.

Some of the error codes the VI Control VIs return are different in LabVIEW 7.0. In previous versions of LabVIEW, the VI Control VIs returned the error codes 7 and 1000. The VI Control VIs in LabVIEW 7.0 return the codes 1004 and 1003. If a VI built in LabVIEW 4.*x* checks for error codes 7 and 1000, you must modify the VI to work in LabVIEW 7.0.

DDE VIs

(Windows) The DDE VIs do not appear on the **Functions** palette but exist as compatibility VIs in the `vi.lib\platform\dde.llb`.

Upgrading from LabVIEW 3.*x* or Earlier Versions

Refer to the National Instruments Web site at `ni.com` for information about using a VI conversion kit to upgrade from LabVIEW 3.*x* or earlier. Refer to the *Upgrading from LabVIEW 4.x*, *Upgrading from LabVIEW 5.x*, and *Upgrading from LabVIEW 6.x* sections of this document for information about other upgrade issues you might encounter.

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between version 3.*x* and 7.0 and to the *LabVIEW 5.1 Addendum* at `ni.com/manuals` for more information about the new features and changes in each version.

LabVIEW 7.0 Features

Refer to the *LabVIEW User Manual* and the *LabVIEW Help* for more information about LabVIEW 7.0 features.

Express VIs

Use the Express VIs for common measurement tasks. Express VIs are nodes that require minimal wiring because you configure them with dialog boxes. Express VIs are located on the **Functions** palette, and they appear on the **Functions** palette with white backgrounds surrounded by a blue border.

Refer to the *Getting Started with LabVIEW* manual for more information about using Express VIs. Refer to the `examples\express` directory for examples of using Express VIs with other built-in VIs and functions.

Dynamic Data Type

The dynamic data type appears as a dark blue terminal, shown at left. Most Express VIs accept or return dynamic data. You can wire dynamic data to any indicator or input that accepts numeric, waveform, or Boolean data. Wire dynamic data to an indicator that can best present the data. Indicators include a graph, chart, or numeric indicator.

In addition to the data associated with a signal, dynamic data includes attributes that describe the signal, such as the name of the signal or the date and time the data was acquired. Attributes specify how the signal appears on a graph or chart. For example, if you use the DAQ Assistant Express VI to acquire a signal and plot that signal on a graph, the name of the signal appears in the plot legend of the graph, and the x-scale adjusts to display timing information associated with the signal.

Refer to the Graph Adapting to Attributes VI in the `examples\general\graphs` directory for an example of a graph that adapts to the attributes of dynamic data.

LabVIEW Dialog Box Enhancements

Use the abbreviated menus in the **LabVIEW** dialog box to access common LabVIEW features and utilities.

Click the **New** button to display the **New** dialog box, which lists built-in VI templates. To create a new, blank VI, select **Blank VI** from the **New** pull-down menu or select **File»New VI**.

Click the **Configure** button to launch Measurement & Automation Explorer (MAX). To configure a new NI-DAQmx channel or task, select the appropriate item from the **Configure** pull-down menu.

Using Template VIs to Create New VIs

Select **File»New** to display the **New** dialog box, which lists built-in template VIs you can use to build common measurement applications. You also can display the **New** dialog box by clicking the **New** button on the **LabVIEW** dialog box.

Controls and Functions Palette Enhancements

LabVIEW includes the following enhancements to the **Controls** and **Functions** palettes:

- To change to another palette view or format, click the **Options** button on the **Controls** or **Functions** palette toolbar. In the **Controls/Functions Palettes** page that appears, you can set the following options:
  - **Palette View**—The Advanced and Express palette views replace the Basic, Data Acquisition, default, and Test & Measurement palette views in LabVIEW 6.1. The default palette view is Express, which includes subpalettes on the top level of the **Controls** and **Functions** palettes that contain Express VIs and other objects you need to build common measurement applications. The **All Controls** and **All Functions** subpalettes in the Express palette view contain the complete set of built-in controls, indicators, VIs, and functions. The Advanced palette view includes subpalettes on the top level that contain the complete set of built-in controls, indicators, VIs, and functions. The **Express** subpalettes in the Advanced palette view contain Express VIs and other objects you need to build common measurement applications.

Note In the Express palette view, toolsets and modules do not install subpalettes on the top level of the **Controls** and **Functions** palettes. Instead, toolsets and modules install subpalettes on the **All Controls** and **All Functions** subpalettes. In the Advanced palette view, toolsets and modules install subpalettes on the top level.

  - **Format**—Available formats include **Icons and Text**, which displays all palette items as icons with text labels below each icon, and **Standard (Icons or Text)**, which displays palette items as icons on the main palette and displays subpalettes in the format of the individual subpalette. If you select the **Standard (Icons or Text)** format and the format of the individual subpalette is **Icons and Text**, LabVIEW displays the items in that subpalette as icons. The **Standard (Icons or Text)** format is the same as the **Standard** format in LabVIEW 6.1 and earlier. The **Standard** format in LabVIEW 7.0 displays all palettes in the format of the individual palette, including the main palette.
  - **Navigation Buttons**—The **Search** button on the palette toolbar has a text label. You can configure LabVIEW to display a text label for all buttons on the palette toolbar or to display no labels for the buttons.

  - **Palette Loading**—LabVIEW loads information about the palettes in the background when you are not using the mouse or keyboard, which improves the performance of LabVIEW when you search for objects on the palettes. You can configure LabVIEW to load the palette information as you navigate the palettes, or you can configure LabVIEW to load the palette information when LabVIEW starts.
  - **Use Window Titles in Functions palette**—This option moved from the **Block Diagram** page of the **Options** dialog box to the **Controls/Functions Palettes** page.
  - **Allow search in temporary palette**—When you right-click the front panel or block diagram to display a temporary version of the **Controls** or **Functions** palette, a **Search** button appears on the temporary palette. Remove the checkmark from this checkbox to configure LabVIEW not to display the **Search** button on temporary palettes.
- Save controls and VIs in the `labview\user.lib` directory to add them to the **User Controls** and **User Libraries** palettes, respectively. Save controls and VIs in the `labview\user.lib\_express` directory to add them to the **Express User Controls** and **Express User Libraries** palettes, respectively.
- To create or edit a custom palette view, select **Tools»Advanced»Edit Palette Views**.
- In custom palette views, you can configure the text labels that appear below each icon when the palette format is **Icons and Text**. To do so, select **Tools»Advanced»Edit Palette Views**, select a custom palette view from the **Palette View** pull-down menu, right-click an icon on the palette, and select **Edit Short Name** from the shortcut menu.
- You can resize the **Controls** and **Functions** palettes the same way you resize the front panel and block diagram windows. Click the **Restore Palette Size** button on the palette toolbar to resize the palette to its default size.
- LabVIEW includes the following enhancements to searching the palettes:
  - While a palette is in search mode, click the **Return to Palette** button to exit search mode and return to the palette. The next time you click the **Search** button, the text box contains the last string you entered in the text box, which is useful if you double-click a search result and you want to return to the search results.
  - LabVIEW displays a light blue glyph to the left of Express VIs in the search results.

  - If two or more objects in the search results have the same name, LabVIEW displays the name of the palette that contains the object in brackets to the right of the object name. For example, if you search for `While Loop`, the search results include **`While Loop <<Execution Control>>`** and **`While Loop <<Structures>>`**.
  - While a palette is in search mode, click the **Options** button at the bottom of the palette to configure LabVIEW to search the palettes by **Name and Keyword** or by **Name**. These options replace the **Begins With** and **Contains** options in LabVIEW 6.1. If you search by **Name and Keyword**, LabVIEW searches for the text you enter in the names of objects on the palettes and in the keywords associated with objects on the palettes. If you search by **Name**, LabVIEW searches only the names of objects on the palettes.
  - When you click the **Search** button on the **Controls** palette, LabVIEW searches only controls by default. When you click the **Search** button on the **Functions** palette, LabVIEW searches only VIs and functions by default. You can expand the search to all objects on the **Controls** and **Functions** palettes by clicking the **More** button.
- Click the **Up** button on the palette toolbar and hold the mouse button down to display a shortcut menu that lists each subpalette in the path to the current subpalette. Select a subpalette name in the shortcut menu to navigate to the subpalette.

Properties Dialog Boxes

Use properties dialog boxes to configure how controls or indicators appear or behave. Right-click a control or indicator on the front panel and select **Properties** from the shortcut menu to display the Properties dialog box for that object.

Automatic Error Handling

By default, LabVIEW automatically handles any error that occurs when a VI runs by suspending execution, highlighting the subVI or function where the error occurred, and displaying an error dialog box.

Select **File»VI Properties**, select **Execution** from the **Category** pull-down menu, and remove the checkmark from the **Enable automatic error handling** checkbox to disable automatic error handling for a VI. Select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and remove the checkmark from the **Enable automatic error handling in new VIs** checkbox to disable automatic error handling for any new VIs you create. Remove the checkmark from the **Enable automatic error**

handling dialogs checkbox if you do not want LabVIEW to suspend execution and display an error dialog box when an error occurs in existing VIs with automatic error handling enabled.

Icons for Front Panel Terminals

You can configure front panel controls or indicators to appear as icon or data type terminals on the block diagram. By default, front panel objects appear on the block diagram as icon terminals. For example, the icon terminal shown at left represents a knob on the front panel. The DBL at the bottom of the terminal represents a data type of double-precision, floating-point numeric.

Right-click a terminal and select **Display Icon** from the shortcut menu to remove the checkmark and to display the data type terminal. To display the icon terminal, select **Display Icon** from the shortcut menu again. Icon terminals are larger than data type terminals, so you might unintentionally obscure other block diagram objects when you convert a data type terminal to an icon terminal.

Select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and remove the checkmark from the **Place front panel terminals as icons** checkbox to configure LabVIEW to display terminals for new front panel objects you create as data types.

Automatic Wire Routing

LabVIEW automatically finds a route for a wire as you wire it. To automatically route an existing wire, right-click the wire and select **Clean Up Wire** from the shortcut menu. Press the <A> key after you start a wire to temporarily disable automatic wire routing and route a wire manually. Press the <A> key again to enable automatic wire routing for the wire. After you end the wire, LabVIEW enables automatic wire routing again. You also can temporarily disable automatic routing after you click to start or set a wire by holding down the mouse button while you wire to another terminal or set point and then release the mouse button. After you release the mouse button, LabVIEW enables automatic wire routing again.

Select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and remove the checkmark from the **Enable automatic wire routing** checkbox to disable automatic wire routing for all new wires.

Resizing Structures Automatically

When you place or move an object in a structure near the structure border, the structure resizes to add space for that object. When you resize a structure manually, you cannot resize it smaller than the objects inside it. To disable the automatic resizing behavior for a structure, right-click the structure border and select **Auto Grow** from the shortcut menu to remove the checkmark. This behavior is enabled by default for all new structures you place on the block diagram. Select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and remove the checkmark from the **Place structures with Auto Grow enabled** checkbox to disable the automatic resizing behavior for all new structures.

Flat Sequence Structure

LabVIEW includes two types of sequence structures— the Flat Sequence structure and the Stacked Sequence structure. The Flat Sequence structure, like the Stacked Sequence structure, contains one or more subdiagrams, or frames, that execute in sequential order. The Flat Sequence structure displays all the frames at once and executes the frames from left to right until the last frame executes. Data in each frame leave after the frame executes, so you can pass data from each frame to nodes outside the Flat Sequence structure. Use the Flat Sequence structure to avoid using sequence locals and to better document the block diagram.

Right-click a Stacked Sequence structure and select **Replace»Replace with Flat Sequence** from the shortcut menu to convert a Stacked Sequence structure to a Flat Sequence structure.

NI Example Finder

Select **Help»Find Examples** to launch NI Example Finder, which you can use to browse or search installed example VIs and example VIs on the NI Developer Zone at `ni.com/zone`.

(Mac and UNIX) You cannot browse or search example VIs on the Web.

Changing the Default Style of Controls and Indicators

Select **File»VI Properties** and select **Editor Options** from the **Category** pull-down menu to change the style of control or indicator LabVIEW creates when you right-click a terminal and select **Create»Control** or **Create»Indicator** from the shortcut menu. Select **Tools»Options** and select **Front Panel** from the top pull-down menu to change the style of control or indicator LabVIEW creates in new VIs when you right-click a terminal and select **Create»Control** or **Create»Indicator** from the shortcut menu.

LabVIEW Data Directory

When you install LabVIEW, the installer creates a `LabVIEW Data` subdirectory in the default file directory for the operating system to help you organize and locate the data files LabVIEW generates. Use this directory to store any file a VI generates, such as `.lvm` files. Select **Tools»Options** and select **Paths** from the top pull-down menu to specify a different default data directory.

Updated NI Application Programming Interfaces

National Instruments has updated several hardware application program interfaces (APIs). In particular, National Instruments has introduced a new DAQ API, NI-DAQmx for Windows. Refer to the *LabVIEW Measurements Manual* and the documentation for each API more information about these updates.

(Windows) Refer to the `examples\DAQmx` directory for examples of using NI-DAQmx VIs.

Refer to the `examples\instr\visa.llb` and the `examples\instr\smplserl.llb` for examples of using VISA VIs and functions.

DAQ Assistant

(Windows) The DAQ Assistant is a graphic interface for configuring NI-DAQmx measurement tasks, channels, and scales for use in LabVIEW 7.0 and later. Use the DAQ Assistant to configure devices when programming with NI-DAQmx. You can launch the DAQ Assistant from the LabVIEW dialog box, the DAQmx Task and DAQmx Global Channel I/O controls, and the DAQ Assistant Express VI.

Instrument I/O Assistant

(Windows) You usually communicate with an instrument with an instrument driver. If a driver does not exist, you can use the Instrument I/O Assistant to communicate with an instrument that uses a serial, Ethernet, or GPIB interface and graphically parse the response. Instrument I/O Assistant organizes instrument communication into ordered steps. Launch Instrument I/O Assistant by placing the I/O Assistant Express VI on the block diagram or by double-clicking the I/O Assistant Express VI icon on the block diagram.

I/O Controls

(Windows) The **I/O** palette includes I/O controls for Motion devices, FieldPoint devices, and NI-DAQmx. Use the FieldPoint I/O point control to access the FieldPoint items you create and configure using MAX. Use the Motion resource control to access a motion device you configure using MAX. Any items you configure in MAX appear as options in the I/O control pull-down menu.

(Windows) Use the controls located on the **DAQmx Name Controls** palette to access the tasks, global channels, physical channels, terminals, scales, devices, and switch resources you configure using MAX and the DAQ Assistant. Refer to the `examples\DAQmx` directory for examples of using NI-DAQmx VIs.

Updates to Instrument Drivers

National Instruments often updates instrument drivers and recommends that you download new versions of the drivers from the Instrument Driver Network at `ni.com/idnet`. The updated instrument drivers are compiled for LabVIEW 7.0 and are installed in the `labview` directory. Updates to the IVI drivers fix a problem that might cause the Initialize, Initialize with Options, and Close functions to report errors incorrectly.

Miscellaneous I/O Enhancements

LabVIEW includes the following enhancements to I/O:

- The VISA Wait on Event function performs I/O operations synchronously or asynchronously. By default, the function handles I/O operations asynchronously. Right-click the node and select **Do I/O Synchronously** from the shortcut menu to wait synchronously for an occurrence of the specified event. In LabVIEW 6.1 and earlier, the VISA Wait on Event synchronously waited for an event while the VISA Wait on Event Async VI asynchronously waited for an event. The VISA Wait on Event function in LabVIEW 7.0 replaces the VISA Wait on Event Async VI in LabVIEW 6.1 and earlier.
- Use the VISA USB VI and functions to control USB devices.
- The ReceiveSetup and SendSetup functions no longer include the **byte count** output.
- The data type of the Recv CIC State property of the GPIB CIC event changed from numeric to Boolean.
- If you use a Property Node to access the DAQmx properties, the Property Node shortcut menu includes a **Change To** submenu. To get property information, right-click the Property Node and select **Change To»Change to Read** from the shortcut menu. To set property information, right-click the Property Node and select

Change To»Change to Write from the shortcut menu. If a property is read only, **Change to Write** appears dimmed in the shortcut menu. To reset a property to its default value, right-click the Property Node and select **Change To»Default Value** from the shortcut menu. If you cannot set a property to its default value, **Default Value** appears dimmed in the shortcut menu.

.NET Functions

Use the .NET functions to create .NET objects and set properties or methods on those objects. You must install the Microsoft .NET framework to use the .NET functions. Refer to the MSDN Web site for more information about .NET and installing the framework. LabVIEW does not support events using delegates or Windows Forms.

Mac OS X Support

LabVIEW 7.0 provides support for Mac OS X. You need to use the LabVIEW Real-Time Module to perform DAQ I/O on Mac OS X. Refer to the *LabVIEW RT Module for Mac OS X User Manual Addendum* for more information about performing I/O with Mac OS X. CINs compiled for LabVIEW on Mac OS 9.*x* or earlier are not compatible with LabVIEW on Mac OS X. Refer to the *Compile on Mac OS X* section of Chapter 3, *CINs*, in the *Using External Code in LabVIEW* manual or the `ReadMe(Project Builder).rtf` file in the `cintools\Project Builder Files` directory for more information about rebuilding CINs for Mac OS X.

LabVIEW Environment Enhancements

The LabVIEW environment includes several enhancements, such as abridged menus, icons on front panel terminals, and automatic tool selection.

Select **Tools»Options** and select **New and Changed in 7.0** from the top pull-down menu to customize these enhancements to the LabVIEW environment. The following sections describe the options in the order they appear on the **New and Changed in 7.0** page.

New Block Diagram Options

- **Place front panel terminals as icons**—Displays terminals for new front panel objects you create as icons. Remove the checkmark from this checkbox to display terminals for new front panel objects as data types. Refer to the *Icons for Front Panel Terminals* section of this document for more information about configuring existing terminals to appear as icons or data types.

- **Enable automatic error handling in new VIs**—For new VIs you create, LabVIEW automatically handles any error that occurs when the VI runs by suspending execution, highlighting the subVI or function where the error occurred, and displaying an error dialog box. Remove the checkmark from this checkbox to disable automatic error handling for all new, blank VIs. Refer to the *Automatic Error Handling* section of this document for more information about disabling automatic error handling for existing VIs.
- **Enable automatic error handling dialogs**—For existing VIs with automatic error handling enabled, LabVIEW automatically handles any error that occurs when the VI runs by suspending execution, highlighting the subVI or function where the error occurred, and displaying an error dialog box. Remove the checkmark from this checkbox if you do not want LabVIEW to suspend execution and display an error dialog box when an error occurs in existing VIs with automatic error handling enabled. Refer to the *Automatic Error Handling* section of this document for more information about disabling automatic error handling for existing VIs.
- **Enable automatic wire routing**—LabVIEW automatically finds a route for wires as you wire them. LabVIEW routes a wire around existing objects on the block diagram, such as loops and structures. LabVIEW also routes a wire to decrease the number of bends in the wire. Remove the checkmark from this checkbox to disable automatic wire routing for all new wires. If you disable automatic wire routing, you can right-click individual wires and select **Clean Up Wire** from the shortcut menu to automatically route them. You can temporarily disable automatic wire routing by pressing the <A> key after you start a wire. Refer to the *Automatic Wire Routing* section of this document for more information about temporarily disabling automatic wire routing.
- **Auto insert Feedback Node in cycles**—Inserts a Feedback Node in a For Loop or While Loop when you wire the output of a subVI, function, or group of subVIs and functions to the input of that same VI, function, or group. Remove the checkmark from this checkbox if you do not want LabVIEW to insert Feedback Nodes automatically. Refer to the *Feedback Nodes* section of this document for more information about Feedback Nodes.
- **Place subVIs as expandable**—Displays subVIs you place on the block diagram as expandable nodes. Use expandable nodes to make wiring easier and to aid in documenting block diagrams. This option does not affect Express VIs. Refer to the *Displaying SubVIs as Expandable Nodes* section of this document for more information about displaying subVIs you already placed on the block diagram as expandable nodes.

- **Configure Express VIs immediately**—Displays a configuration dialog box immediately after you place an Express VI on the block diagram. Remove the checkmark from this checkbox if you do not want LabVIEW to display configuration dialog boxes immediately after you place Express VIs on the block diagram. Refer to the *Express VIs* section of this document for more information about Express VIs.
- **Place structures with Auto Grow enabled**—For new structures you place on the block diagram, the structure resizes to add space for any objects you place or move in the structure near the structure border. Remove the checkmark from this checkbox to disable the automatic resizing behavior for all new structures. Refer to the *Resizing Structures Automatically* section of this document for more information about enabling the automatic resizing behavior for existing structures.
- **Show red Xs on broken wires**—Displays broken wires as dashed black lines with red Xs in the middle. Remove the checkmark from this checkbox to display broken wires as dashed black lines without red Xs in the middle. Refer to the *Broken Wire Enhancements* section of this document for more information about the appearance of broken wires.

New Front Panel Options

- **Control style for new VIs**—Changes the style of control or indicator LabVIEW creates in new VIs when you right-click a terminal and select **Create»Control** or **Create»Indicator** from the shortcut menu. Refer to the *Changing the Default Style of Controls and Indicators* section of this document for more information about changing the style of controls or indicators LabVIEW creates in existing VIs.

New Miscellaneous Options

- **Use abridged menus**—**(Windows and UNIX)** Displays only the most common and most recently used menu items by default. Click the arrows at the bottom of a menu to display all items. Remove the checkmark from this checkbox to display all menu items by default.
- **Lock automatic tool selection on**—Keeps automatic tool selection enabled when you press the <Tab> or <Shift-Tab> keys. Remove the checkmark from this checkbox to configure LabVIEW to disable automatic tool selection when you press the <Tab> key and toggle automatic tool selection when you press the <Shift-Tab> keys. Refer to the *Automatic Tool Selection Enhancements* section of this document for more information about enabling and disabling automatic tool selection.

- **Enable Just-In-Time Advice**—Displays the **Just-In-Time Advice** window when you perform certain actions in LabVIEW. The dialog box alerts you to changes between a previous version of LabVIEW and the current version of LabVIEW. Remove the checkmark from this checkbox to disable the **Just-In-Time Advice** window so it does not appear while you use LabVIEW.
- **Enable Windows Explorer for LLB files**—**(Windows 2000/XP/Me/98)** Enables Windows Explorer support for `.llb` files so double-clicking an `.llb` file displays the contents of the library. You can open, move, copy, rename, and delete files in the library, and you can mark a VI in the library as a top-level VI by right-clicking the VI and selecting **Top Level** from the shortcut menu. If you remove the checkmark from this checkbox, LabVIEW opens all the top-level VIs in the library when you double-click an `.llb` file.

Changed Defaults Since 6.1

- **Delete/copy panel terminals from diagram**—Allows you to delete front panel objects by selecting their terminals on the block diagram and pressing the <Delete> key. This option also allows you to copy front panel terminals on the block diagram by pressing the <Ctrl> key while you drag the terminal to a new location on the block diagram. **(Mac OS)** Press the <Option> key. **(Sun)** Press the <Meta> key. **(Linux)** Press the <Alt> key.

  Remove the checkmark from this checkbox if you do not want to be able to delete or copy front panel terminals on the block diagram.
- **Show dots at wire junctions**—Displays a dot where a wire branches, helping to differentiate between a wire junction and the crossing of unconnected wires. Remove the checkmark from this checkbox to display wire junctions without dots.

New Pages in the Options Dialog Box

- **Alignment Grid**—Use this page to set grid options for all new VIs, including the size, contrast, and style of the grid. Refer to the *Aligning Objects Using the Alignment Grid* section of this document for more information about using the alignment grid.
- **Controls/Functions Palettes**—Use this page to select a palette view or change its format. Refer to the *Controls and Functions Palette Enhancements* section of this document for more information about customizing the **Controls** and **Functions** palettes.

Aligning Objects Using the Alignment Grid

Select **Operate»Enable Alignment Grid on Panel** to enable the alignment grid on the front panel and align objects as you place them. Select **Operate»Disable Alignment Grid on Panel** to disable the alignment grid and use the visible grid to align objects manually. You also can press the <Ctrl-#> keys to enable or disable the alignment grid. On French keyboards, press the <Ctrl-”> keys to enable or disable the alignment grid.

(Mac OS) Press the <Command-*> keys. **(Sun)** Press the <Meta-#> keys. **(Linux)** Press the <Alt-#> keys.

You also can use the alignment grid on the block diagram.

To set grid options for all new VIs, select **Tools»Options** and select **Alignment Grid** from the top pull-down menu. To set the size of the grid for the current VI, select **File»VI Properties** and select **Editor Options** from the **Category** pull-down menu.

Automatic Tool Selection Enhancements

LabVIEW includes the following enhancements to automatic tool selection:

- If you want to use the <Tab> key to cycle through the four most common tools on the **Tools** palette, click the **Automatic Tool Selection** button on the **Tools** palette to disable automatic tool selection. Press the <Shift-Tab> keys or click the **Automatic Tool Selection** button to enable automatic tool selection again. You also can manually select a tool on the **Tools** palette to disable automatic tool selection. Press the <Tab> or <Shift-Tab> keys or click the **Automatic Tool Selection** button on the **Tools** palette to enable automatic tool selection again. If automatic tool selection is disabled, you can press the spacebar to switch to the next most useful tool.
- You can configure LabVIEW to disable automatic tool selection when you press the <Tab> key and to toggle automatic tool selection when you press the <Shift-Tab> keys. To do so, select **Tools»Options**, select **Miscellaneous** from the top pull-down menu, and remove the checkmark from the **Lock automatic tool selection on** checkbox.
- To add a marker to a numeric object with arbitrary marker spacing, press the <Ctrl> key and drag an existing marker to the location where you want to add a marker. On slide controls and indicators, you also can drag one of the end markers to the location where you want to add a marker. On rotary controls and indicators, press the <Ctrl> key and drag one of the end markers to the location where you want to add a marker. **(Mac OS)** Press the <Option> key. **(Sun)** Press the <Meta> key. **(Linux)** Press the <Alt> key.

- LabVIEW selects the Positioning tool in the **Hierarchy** window. Press the <Ctrl-Shift> keys when the cursor is not over an icon in the window to switch to the Scrolling tool. **(Mac OS)** Press the <Option-Shift> keys. **(Sun)** Press the <Meta-Shift> keys. **(Linux)** Press the <Alt-Shift> keys.
- Double-click any open space on the front panel or block diagram to create a free label.
- Automatic tool selection is available on the block diagram while a VI runs.

Editing and Customizing Unit Labels

To edit a unit label, right-click it and select **Build Unit String** from the shortcut menu to display the **Build Unit String** dialog box. Use the dialog box to apply a prefix or exponent to a unit, place a unit in the denominator, or add or remove units from the unit string. The dialog box lists units available in LabVIEW in alphabetical order or in groups of unit type.

You can right-click a unit label to toggle autosizing or rotate the label similar to how you customize other front panel labels. You can use Property Nodes to read and write Unit Label properties for numeric objects, graph scales, and color graph scales similar to how you read and write other front panel label properties.

Refer to the Programmatic Units VI in the `examples\general\controls\numerics.llb` for an example of writing Unit Label properties.

Saving VIs for Use in Previous Versions

In LabVIEW 7.0, you can save VIs for use in LabVIEW 6.1 by selecting **File»Save with Options** and selecting the **Save for Previous** option. If you want to save a VI for use in LabVIEW 6.0, you must open the VI in LabVIEW 6.1 and save it by selecting **File»Save with Options** and selecting the **Save for Previous** option.

Printing and Report Generation Enhancements

LabVIEW includes the following enhancements to printing and generating reports:

Note The VIs described in this section are available only in the LabVIEW Full and Professional Development Systems.

- Use the Easy Print VI Panel or Documentation VI to print a front panel or VI documentation or to save the front panel or VI documentation to a report. This VI replaces the Print Panel VI, which no longer appears on the **Functions** palette.

- Use the VI Documentation VIs to customize the VI documentation you print or save to a report.
- **(Windows)** Use the Query Available Printers VI to list the printers available on the computer, including the default printer in LabVIEW.
- The Application and VI classes contain several new VI Server properties you can use to set printing options programmatically.
- **(Windows)** Use the Print Report VI to print an HTML report to the default printer.
- Use the **HTML footer size** input of the Set Report Footer Text VI to set the HTML heading tag to use for the footer in a report.
- The Append Front Panel Image to Report VI is polymorphic. You can wire a path, VI reference, or string to the VI to specify which front panel to append to the report. The old string instance of the Append Front Panel Image to Report VI is similar to the Append Front Panel Image to Report VI in LabVIEW 6.1 and earlier.
- The Append Table to Report VI is polymorphic and has instances that replace the Append Text Table to Report VI and the Append Numeric Table to Report VI in LabVIEW 6.1 and earlier.
- The **Print Options** page of the **VI Properties** dialog box contains a **Scale printed block diagram to fit page** checkbox. Place a checkmark in this checkbox to scale the block diagram to fit on a printed page.
- If you use the Print VI To Printer method in a stand-alone application or shared library, LabVIEW prints only the front panel.
- The **Surround panel with border** checkbox in the **Print** dialog box moved from the **Custom details** page to the **Printer** page.
- In the **Print** dialog box, the **Ordered (Repeat from higher level if nested)** checkbox replaces the **Repeat upper-level frame(s)** checkbox.

Miscellaneous Environment Enhancements

LabVIEW includes the following enhancements to the LabVIEW environment:

- The **Window»Show Panel** and **Show Diagram** menu items changed to **Window»Show Front Panel** and **Show Block Diagram**, respectively.
- The front panel window title is ***This.vi*** **Front Panel**, and the block diagram window title is ***This.vi*** **Block Diagram** where ***This.vi*** is the name of the VI.
- Select **Tools»Find VIs on Disk** to search for a VI by filename in a directory.

- If you receive an internal LabVIEW error, such as `panel.cpp, line 2978`, LabVIEW saves the error information to a log file in the `lvfailurelog` subdirectory of the default data directory. The next time you start LabVIEW, a dialog box prompts you to investigate the error. You can investigate the error at that time, or you can investigate it at a later time by selecting **Help»Investigate Internal Error**. If you investigate an error, LabVIEW sends the error information from the selected log file to the National Instruments Web site and displays the Web site in the default Web browser. The form that appears in the browser contains the error information so you can search for possible causes of the error.
- **(Windows XP)** LabVIEW uses the current Windows XP theme to display objects such as dialog controls, scroll bars, menus, and so on. When you distribute a LabVIEW-built application on Windows XP, the application also uses the current Windows XP theme to display objects.
- When you move the Positioning tool over a resizable object, resizing handles appear at the points where you can resize the object. To restrict the growth vertically or horizontally, use the resizing handles at the top or bottom or on the sides of the object, respectively.
- Press the <Ctrl-Enter> keys to end text entry in front panel and block diagram objects.

Using Tree Controls

Use the tree control to give users a hierarchical list of items from which to select. You organize the items you enter in the tree control into groups of items, or nodes. Click the expand symbol next to a node to expand it and display all the items in that node. You also can click the symbol next to the node to collapse the node.

Note You can create and edit tree controls only in the LabVIEW Full and Professional Development Systems. If a VI contains a tree control, you can run the VI in all LabVIEW packages, but you cannot configure the control in the Base Package.

You configure items in a tree control the same way you configure them in a listbox. You also can change the type of symbol that appears next to each node, and you can configure whether the user can drag items within the tree control.

You also can use the TreeControl properties and methods to configure a tree control programmatically. Use the TreeControl events to handle events the user generates on the tree control.

Refer to the Directory Hierarchy in Tree Control VI in the `examples\general\controls\Directory Tree Control.llb` for an example of using a tree control.

Using Subpanel Controls

Use the subpanel control to display the front panel of another VI on the front panel of the current VI. For example, you can use a subpanel control to design a user interface that behaves like a wizard. Place the **Back** and **Next** buttons on the front panel of the top-level VI and use a subpanel control to load different front panels for each step of the wizard.

Note You can create and edit subpanel controls only in the LabVIEW Full and Professional Development Systems. If a VI contains a subpanel control, you can run the VI in all LabVIEW packages, but you cannot configure the control in the Base Package.

When you place a subpanel control on the front panel, LabVIEW does not create a front panel terminal for the control on the block diagram. Instead, LabVIEW creates an Invoke Node on the block diagram with the Insert VI method selected.

Use the SubPanel properties and methods to modify subpanel controls programmatically.

Refer to the `examples\general\controls\subpanel.llb` for examples of using subpanel controls.

Ring Control Enhancements

Right-click a ring control and select **Edit Items** from the shortcut menu to add items to the list from which you can select in the control. You also can configure the ring control so users can enter numeric values not already associated with any entries in the list of items defined for the control by right-clicking the ring control and selecting **Allow Undefined Values** from the shortcut menu.

To enter an undefined value in the ring control at run time, click the control, select **<Other>** from the shortcut menu, enter a numeric value in the digital display that appears, and press the <Enter> key.

When you configure the list of items for a ring control, you can assign a specific numeric value to each item. If you do not assign specific numeric values to the items, LabVIEW assigns sequential values that correspond to the order of the items in the list, starting with a value of `0` for the first item. To assign specific numeric values, right-click the ring control, select **Edit Items** from the shortcut menu, and remove the checkmark from the

Sequential values checkbox in the **Edit Items** page of the **Ring Properties** dialog box. In the **Values** section of the table in this dialog box, change the numeric values that correspond to each item in the control. Each item in the ring control must have a unique numeric value.

Note You cannot allow the user to enter undefined values in enumerated type controls, and you cannot assign specific numeric values to items in enumerated type controls.

Refer to the Ring and Combo Box VI in the `examples\general\controls\ring.llb` for an example of using ring controls and combo boxes.

Using Combo Box Controls

Use the combo box control to create a list of strings you can cycle through on the front panel. A combo box control is similar to a text or menu ring control. However, the value and data type of a combo box control are strings instead of numbers as with ring controls.

Right-click a combo box control and select **Edit Items** from the shortcut menu to add strings to the list from which you can select in the control. By default, the combo box control allows users to enter string values not already in the list of strings defined for the control. Right-click the combo box control and select **Allow Undefined Strings** from the shortcut menu to remove the checkmark and prevent the user from entering undefined string values in the control.

As you type a string in a combo box control at run time, LabVIEW selects the first, shortest string in the control that begins with the letters you type.

When you configure the list of strings for a combo box control, you can specify a custom value for each string, which is useful if you want the string that appears in the combo box control on the front panel to be different than the string value the combo box terminal on the block diagram returns. Right-click the combo box control, select **Edit Items** from the shortcut menu, and remove the checkmark from the **Values match labels** checkbox in the **Edit Items** page of the **Combo Box Properties** dialog box. In the **Values** column of the table in this dialog box, change the value that corresponds to each string in the control.

Refer to the Ring and Combo Box VI in the `examples\general\controls\ring.llb` for an example of using ring controls and combo boxes.

Graph and Chart Enhancements

LabVIEW includes the following enhancements to graphs and charts:

- Right-click a graph and select **Visible Items»X Scrollbar** from the shortcut menu to display a scroll bar below the graph. Use the Operating tool to click the scroll bar and scroll along the x-axis. Use the X Scrollbar Visible property to display the x-axis scrollbar on a graph or chart programmatically. This property replaces the Scrollbar Visible properties in the IntensityChart and WaveformChart classes.
- Right-click a graph or chart and select **Advanced»Reset Scale Layout** to return the y-scale to the left of the plot area, to return the x-scale to the bottom of the plot area, and to reset the scale markers.
- You cannot duplicate x-scales on charts.
- The Zoom to Fit tool on the graph palette replaces the Undo Zoom tool. Use the Zoom to Fit tool to autoscale all x- and y-scales on a graph or chart.
- By default, graph and chart plot legends automatically resize to the width of the longest plot name visible in the legend. To disable this behavior, right-click the graph or chart and select **Autosize Legend** from the shortcut menu to remove the checkmark. You also can use the Autosize Legend properties in the WaveformGraph and WaveformChart classes to control this behavior programmatically.
- In graphs and charts, the x-axis is configured to use floating-point notation and the y-axis is configured to use automatic formatting. Right-click the graph or chart and select **Properties** from the shortcut menu to display the **Properties** dialog box and configure the formatting of the axes.

Running VIs as Floaters

- You can configure a front panel as a floating window so the window stays on top of all other non-floating LabVIEW windows. To do so, select **File»VI Properties**, select **Window Appearance** from the **Category** pull-down menu, click the **Customize** button, and select the **Floating** option in the **Window Behavior** section. You also can use the Front Panel Window:Behavior property to configure a front panel as a floating window programmatically. This property replaces the Front Panel Window:Is Dialog property.
- Use the Open FP and Close FP methods to open and close a front panel, respectively. Use the Open FP method instead of the Front Panel Window:Open property if you want to set the state of the front panel window when you open it, such as if you want to minimize or maximize the window. National Instruments recommends you use the Open FP method instead of the Front Panel Window:Open property. You also can use the Front Panel Window:State property to set the state

of a front panel window that is already open. Refer to the FrontPanelStates VI in `examples\viserver` for an example of using these properties and methods.

Miscellaneous Front Panel Enhancements

LabVIEW also includes the following front panel enhancements:

- Right-click a numeric control and select **Visible Items»Increment/Decrement** from the shortcut menu to display or hide the increment and decrement buttons. You also can use the Increment/Decrement Visible? property to display or hide the buttons programmatically.
- **(Windows, Mac OS X, and UNIX)** In a multiple-selection listbox, press the <Ctrl> key and click additional items to select more than one item. Press the <Shift> key and click an item above or below the current item to select all items between the current item and the second item you selected. **(Mac OS 9.*x* or earlier)** In a multiple-selection listbox, press the <Shift> key and click additional items to select more than one item.
- **(Windows, Mac OS X, and UNIX)** In a listbox with a selection mode that allows you to select 0 items, press the <Ctrl> key and click the current item to deselect it. **(Mac OS 9.*x* or earlier)** Press the <Shift> key and click the current item.
- Use the **Resize Objects** pull-down menu on the front panel toolbar to resize multiple front panel objects to the same size. You can resize all the selected objects to the width or height of the largest or smallest object, and you can resize all the selected objects to a specific size in pixels.
- Right-click an ActiveX container and select **Advanced»Design Mode** from the shortcut menu to display the container in design mode while you edit the VI. In design mode, events are not generated and event procedures do not run. The default mode is run mode, which means you interact with the object as a user would.
- The **Dialog Controls** palette contains new controls and indicators designed specifically for use in dialog boxes, including a spin control, vertical and horizontal slide controls, and vertical and horizontal progress bars. You cannot right-click these controls and indicators and select **Advanced»Customize** from the shortcut menu to customize their appearance because the controls and indicators use platform-specific drawing functions that do not support customization. You also cannot use Property Nodes and Invoke Nodes to customize the appearance of these controls and indicators.

  The **Dialog Controls** palette also contains a path control, a table, a tree control, and an opaque label that automatically matches the background color of its parent.

- To rotate the scale on a rotary control or indicator, you can drag one of the end markers.
- The **ActiveX** subpalette of the **Controls** palette no longer exists. All ActiveX objects are located on other subpalettes.
- **(UNIX)** When editing any control with text in it, such as a text label, you can use the middle mouse button to paste the most recently highlighted text without having to copy or cut the text first.
- If you paste text into LabVIEW at run time, LabVIEW does not accept the font characteristics of the text you paste. For example, if you paste red text into a string control that contains black text, the text you paste appears in the string control as black text.

Registering Events Dynamically and Handling User Events

In previous versions of LabVIEW, you registered all events statically. In LabVIEW 7.0, you can register events dynamically and create and name custom events.

Use dynamic registration if you want event generation to occur during only part of the application, rather than during the entire run of the top-level VI, or if you want to change which VIs or controls generate events during the run of the application. Dynamic registration also allows you to perform event handling in a subVI rather than only in the VI where the events are generated.

You can create and name events, called user events, which carry user-defined data. Like queues and notifiers, user events allow different parts of an application to communicate asynchronously. However, user events also are integrated with the Event structure, so you can handle both user interface and programmatically generated events in the same Event structure. Use the Value (Signaling) property to update the value of an object and generate an event.

Refer to the `examples\general\dynamicevents.llb` for examples of registering events dynamically and handling user events.

Displaying SubVIs as Expandable Nodes

Use expandable nodes to make wiring easier and to aid in documenting block diagrams. You can display a subVI as an expandable node by right-clicking the subVI and selecting **View As Icon** from the shortcut menu to remove the checkmark.

Select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and place a checkmark in the **Place subVIs as expandable** checkbox to place subVIs on the block diagram as expandable nodes by default.

Broken Wire Enhancements

LabVIEW includes the following enhancements to broken wires:

- A broken wire appears as a dashed black line with a red X in the middle. The arrows on either side of the red X indicate the direction of the data flow, and the color of the arrows indicate the data type of the data flowing through the wire. Move the Wiring tool over a broken wire to display information about why the wire is broken in the **Context Help** window. Right-click the wire and select from shortcut menu options such as **Delete Wire Branch**, **Create Wire Branch**, **Remove Loose Ends**, **Clean Up Wire**, **Change to Control**, **Change to Indicator**, **Enable Indexing at Source**, and **Disable Indexing at Source** to correct the broken wire. These options change depending on the reason for the broken wire. Select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and remove the checkmark from the **Show red Xs on broken wires** checkbox to display broken wires as dashed black lines without red Xs in the middle.
- If wiring to a terminal would create a broken wire, the cursor changes to a wire spool with a warning symbol. You can create the broken wire, but you must correct the broken wire before you can run the VI.

Polymorphic VI Enhancements

You can use the polymorphic VI selector to select the instance of a polymorphic VI manually. Right-click the polymorphic VI on the block diagram and select **Visible Items»Polymorphic VI Selector** from the shortcut menu to display the selector. Use the Operating tool to click the polymorphic VI selector and select an instance from the shortcut menu. For built-in polymorphic VIs, the **Context Help** window displays a description of each instance as you move the cursor through the shortcut menus of the polymorphic VI or its selector. You also can right-click the selector and select **Find Polymorphic VI** from the shortcut menu to find the polymorphic VI node associated with a polymorphic VI selector.

Display the **Polymorphic VI** window for polymorphic VIs you create to configure whether to display the polymorphic VI or instance VI icon on the block diagram and to display the connector pane and description of the polymorphic VI or the instance VI in the **Context Help** window when you move the cursor over the polymorphic VI. You also can configure whether to display the polymorphic VI selector by default when you place the polymorphic VI on the block diagram, and you can configure whether to display the **Automatic** item in the shortcut menus of the polymorphic VI and its selector. Click the **Edit Name** button to edit the name of an instance in the shortcut menus of a polymorphic VI and its selector.

Time Stamp Control and Data Type

Use the time stamp control to use, view, and store absolute time with high precision. The time stamp data type, shown at left, can accurately hold 15 digits of precision in the whole second and 15 digits of precision in the fractions of a second. The time stamp data type is a signed 128-bit fixed-point number with a 64-bit radix. You can interpret the most-significant 64 bits as a 64-bit signed integer. This integer represents the number of whole seconds that have elapsed since 12:00 a.m., January 1, 1904, Universal Time. This integer can hold 2^{63} number of seconds in the future since 1904. You can interpret the least-significant 64 bits as a 64-bit unsigned integer. This integer represents the number of 2^{-64} seconds after the whole seconds you specified in the most-significant 64 bits. That is, the most-significant 64 bits represent seconds, and the least-significant 64 bits represent the fractions of a second. This 128-bit data type represents absolute time composed of the two 64-bit components. You can use a numeric control to display time stamp values. However, the numeric control holds a relative quantity, whereas the time stamp control holds an absolute quantity.

If you load a LabVIEW 6.1 VI in LabVIEW 7.0 that uses the Get Date/Time In Seconds or Date/Time To Seconds functions, LabVIEW inserts the appropriate conversion function on the block diagram to convert the time stamp data to the data type that the LabVIEW 6.1 VI expects.

Note The **Options** dialog box no longer contains a **Time and Date** page. LabVIEW obtains the global setting for time and date from the operating system. If you want to set the time and date for a time stamp control, click the time/date browse button, shown at left, to the right of the control.

Supplied and Custom Probes

Use the supplied probes to create new probes to debug applications. Supplied probes are VIs that display comprehensive information about the data that pass through a wire. For example, the VI Refnum Probe returns information about the VI name, the VI path, and the hex value of the reference. You also can use a supplied probe to respond based on the data that flow through the wire. For example, use an error probe on an error cluster to receive the status, code, and description of the error and specify if you want to pause the VI if an error or warning occurs or if you want to pause the VI when it receives a particular error code. Right-click a wire and select **Custom Probe** from the shortcut menu to select a supplied probe. Only the probes that match the data type of the wire you right-click appear on the shortcut menu. After you select a supplied probe to use, that probe becomes the default probe for that data type, and LabVIEW loads that probe when you right-click the wire and select **Probe** from the shortcut menu.

Refer to the Using Supplied Probes VI in the `examples\general\probles.llb` for an example of using the supplied probes.

Create a custom probe when you want to have more control over how LabVIEW probes the data that flow through a wire. You can create a custom probe based on an existing probe or you can create a new probe. Right-click a wire and select **Custom Probe»New** from the shortcut menu to display the **Create New Probe** window. The Create New Probe window guides you through creating a custom probe based on the data type of the wire you right-click.

Formatting Numbers

By default, LabVIEW displays and stores numbers like a calculator. A numeric control or indicator displays up to six digits before automatically switching to exponential notation. You can configure the number of digits LabVIEW displays before switching to exponential notation in the **Format and Precision** page of the **Properties** dialog box.

You also can use percent code formatting to specify how you want LabVIEW to display numbers.

Vaild Formats of Numeric Data in Waveforms

The data (Y) component of a waveform can be an array of any numeric data type, including integers. The interior of the waveform datatype terminal appears orange if the data consist of floating-point or complex numbers and blue if the data consist of integers.

Picture Control Enhancements

LabVIEW includes the following enhancements to functionality for modifying images:

- In the following VIs, several parameters were replaced with an **image data** cluster parameter: Draw Flattened Pixmap, Flatten Pixmap, Read BMP File, Read JPEG File, Read PNG File, Unflatten Pixmap, Write BMP File, Write JPEG File, and Write PNG File.
- The following VIs are new:
  - Use the Create Mask VI to apply a mask to an image. This VI is useful if you want to make a certain color in an image transparent before writing the image to a picture indicator.
  - Use the Picture to Pixmap VI to convert a picture data type into the **image data** cluster so you can perform certain tasks with the image, such as save it to a file.
  - Use the Get Image Subset VI to obtain a subset of a source image instead of the entire image.

- The Flatten Pixmap VI has a **mask** input, which you can use to apply a mask to a pixmap, and a **colors** input, which you can use to define the color table for the pixmap.
- The Unflatten Pixmap VI has a **mask** output, which describes mask information for each pixel, and a **colors** output, which describes the color table for the pixmap.
- The Draw Unflattened Pixmap VI replaces the Draw 1-bit Pixmap, Draw 4-bit Pixmap, Draw 8-bit Pixmap, and Draw True-Color Pixmap VIs in LabVIEW 6.1. The Draw Unflattened Pixmap VI has a **mask** input, which you can use to apply a mask to the picture.
- In the following VIs, the **pen** input replaces the **black in B/W?** input: Draw Arc, Draw Circle by Radius, Draw Line, Draw Multiple Lines, Draw Oval, Draw Point, Draw Rect, and Draw Round Rect. Use the **pen** input to set the width and style of the pen LabVIEW uses to draw the picture.
- In the Draw Text in Rect and Draw Text at Point VIs, the **Text Orientation** input replaces the **black in B/W?** input. Use the **Text Orientation** input to set the orientation of the text you draw.
- The Get Text Rect VI has a **Text Orientation** input, which you use to indicate the orientation of the text before the VI obtains the bounding rectangle of the text.
- In the following VIs, if you do not wire a path to the VI, LabVIEW displays a file dialog box so the user can navigate to the file: Write BMP File, Write JPEG File, and Write PNG File.
- The following VI Server methods use the **image data** cluster: Get Panel Image (VI), Get VI Icon as Image Data (VI), Set VI Icon from Image Data (VI), Get Image (Control), and Get Term Image (Control).
- The Set VI Icon method was renamed to Set VI Icon from File.
- The Picture class contains several new VI Server properties you can use to modify images programmatically.
- The Read JPEG File VI reads JPEG files as 24-bit images.

Refer to the `examples\picture\pictctl.llb` for examples of these enhancements.

Changing the Cursor Appearance in a VI

Use the Cursor VIs to change the appearance of the cursor on the front panel of a VI. For example, if the VI is acquiring or analyzing data and cannot accept user input, you can change the cursor to an hourglass or watch cursor. After the VI finishes acquiring or analyzing data and can accept user input, you can change the cursor back to the default cursor. You can change the cursor to a system or LabVIEW cursor, and you can create a cursor from a file.

Note The Cursor VIs are available only in the LabVIEW Full and Professional Development Systems.

Refer to the `examples\cursors\CursorUtilities.llb` for examples of changing the appearance of the cursor.

Digital Waveform Data Type and Digital Data Type

Use the digital waveform data type to represent a digital waveform. Use the digital data type to represent digital data in a digital waveform. Wire a digital waveform to a digital data indicator to view the samples and signals of a digital waveform.

Use the Digital Waveform VIs and functions to perform operations on digital waveforms and digital data. The digital data indicator displays digital data arranged in rows and columns. Use the digital data indicator to build digital waveforms or to display digital data extracted from a digital waveform. The digital waveform data type carries the data, start time, delta *x*, and the attributes of a digital waveform. You can use the Build Waveform function located on the **Digital Waveform** palette to create digital waveforms. When you wire a digital waveform to the digital waveform graph, the graph automatically plots a waveform based on the timing information and the digital waveform data.

Feedback Nodes

LabVIEW inserts a Feedback Node in a For Loop or While Loop when you wire the output of a subVI, function, or group of subVIs and functions to the input of that same VI, function, or group. Like a shift register, the Feedback Node stores data when the loop completes an iteration, sends that value to the next iteration of the loop, and transfers any data type. Use the Feedback Node to avoid unnecessarily long wires in loops. The Feedback Node arrow indicates in which direction the data flows along the wire.

You also can select a Feedback Node on the **Functions** palette and place it in a For Loop or While Loop.

If you do not want LabVIEW to insert Feedback Nodes automatically, select **Tools»Options**, select **Block Diagram** from the top pull-down menu, and remove the checkmark from the **Auto insert Feedback Node in cycles** checkbox.

Refer to the Feedback Node Build Array VI in the `examples\general\structs.llb` for an example of using the Feedback Node.

Replacing Shift Registers with Tunnels in Loops

Replace shift registers with tunnels when you no longer need to transfer values from one loop iteration to the next by right-clicking the shift register and selecting **Replace with Tunnels** from the shortcut menu. Replace tunnels with shift registers when you want to transfer values from one loop iteration to the next by right-clicking the tunnel and selecting **Replace with Shift Register** from the shortcut menu.

Miscellaneous Block Diagram Enhancements

LabVIEW also includes the following block diagram enhancements:

- Strictly typed control refnums accept only control refnums of exactly the same data type.
- Right-click any block diagram object and select ***Source*** **Palette** from the shortcut menu to access similar objects from a subpalette, where ***Source*** is the name of the subpalette that contains the block diagram object. If you right-click a node terminal, two subpalette shortcut menu items might appear. The first is the source palette of the node, and the second is the most commonly used source palette of the terminal data type.
- If you do not wire the recommended terminals of a subVI, LabVIEW does not generate any warnings for the VI that contains the subVI.
- Right-click a Case structure, Flat Sequence structure, Stacked Sequence structure, For Loop, or While Loop and select a **Replace** item from the shortcut menu to replace the structure with another similar structure.
- You can create a picture or refnum constant by dragging a front panel picture or refnum control to the block diagram. LabVIEW creates a constant with the value of the front panel control at the time you dragged it to the block diagram. The value of a refnum constant is Not A Refnum. You also can create a picture or refnum constant by right-clicking a block diagram terminal and selecting **Create» Constant** from the shortcut menu. If you create a constant from a control refnum control or terminal, LabVIEW creates a class specifier constant.

- Right-click a wire branch and select **Delete Wire Branch** from the shortcut menu to delete the wire branch.
- In a string constant, press the <Shift-Enter> keys to disable autosizing if it is enabled. If autosizing is disabled, press the <Shift-Enter> keys to display a scroll bar in the constant. If autosizing is disabled and the scroll bar is not visible, the constant resizes vertically as you enter text in the constant. Resizing a string constant smaller than its contents displays the scroll bar. If the scroll bar is visible and you resize the constant larger than its contents, LabVIEW hides the scroll bar.
- The **Functions** palette contains a **Decorations** subpalette.
- When you right-click a Property Node or Invoke Node, the shortcut menu items **Select VI Server Class**, **Select VISA Class**, **Select ActiveX Class**, and so on appear under a top-level **Select Class** menu.

Emailing Data from VIs

Use the SMTP E-mail VIs to send electronic mail, including attached data and files, using the Simple Mail Transfer Protocol (SMTP).

Refer to the `examples\comm\smtpex.llb` for examples of using the SMTP E-mail VIs.

DataSocket Enhancements

LabVIEW includes the following DataSocket enhancements:

- If you want to read all the values published to the DataSocket Server instead of only the most recent, you must buffer the data on the client. You also must use the DataSocket Server Manager to configure server-side buffering. Refer to the *NI DataSocket Server Help* for more information about server-side buffering.
- Use the DataSocket Open and DataSocket Close functions to open or close a DataSocket connection programmatically. Use the DataSocket properties, such as Buffer Maximum Bytes, Buffer Maximum Packets, Buffer Utilization (Bytes), Buffer Utilization (Packets), Connection Status, and URL to configure DataSocket buffering constraints, verify the DataSocket connection status, and read a DataSocket URL.

  Use the **ms timeout (0)** input of the DataSocket Write and Close functions to specify the number of milliseconds the function waits for the pending operation to complete.

UDP Enhancements

Use the UDP Multicast Open VI instead of the UDP Open function to open connections capable of reading, writing, or reading and writing UDP multicast data. A multicast IP address defines a multicast group. Multicast IP addresses are in the `224.0.0.0` to `239.255.255.255` range. When a client wants to join a multicast group, it subscribes to the multicast IP address of the group. Once subscribed to a multicast group, the client receives data sent to the multicast IP address.

Refer to the *Using LabVIEW with TCP/IP and UDP* Application Note for more information about using UDP multicasting. Refer to the UDP Multicast Receiver VI and UDP Multicast Sender VI in the `examples\comm\UDP.llb` for examples of UDP multicasting.

Handling ActiveX Events

The ActiveX Event VIs do not appear on the **Functions** palette. To use ActiveX events in an application, you must use the Register Event Callback node to register and handle ActiveX events.

Refer to the `examples\comm\axevent.llb` for examples of using the Register Event Callback node.

Miscellaneous Communication, VI Server, and Remote Front Panel Enhancements

LabVIEW includes the following enhancements to the VI Server and remote front panels:

- Use the Connection Info and Check Connection Application methods to check if a VI Server connection is responsive. If you do not use these methods, LabVIEW does not check the connection, and the behavior is the same as in previous versions of LabVIEW.

  Refer to the `examples\viserver\connpolling.llb` for examples of using the Connection Info and Check Connection methods.
- The LabVIEW 6.1 and 7.0 browser plug-ins can run concurrently, but they communicate with different remote front panel servers. The LabVIEW 6.1 browser plug-in can display only LabVIEW 6.1 VIs, and the LabVIEW 7.0 browser plug-in can display only LabVIEW 7.0 VIs. The HTML `OBJECT/EMBED` tag determines which plug-in the browser loads. For the LabVIEW ActiveX control, the `CLASSID` you specify in the `OBJECT` tag determines which plug-in to load. The `CLASSID` of the LabVIEW 6.1 browser plug-in and the `CLASSID` of the LabVIEW 7.0 browser plug-in are different. For the Netscape browser plug-in, the MIME type you specify in the `EMBED` tag determines which plug-in to load. Refer to the *LabVIEW Help* in

LabVIEW 6.1 for more information about the syntax used in LabVIEW 6.1.

- You can export data-intensive VIs for remote viewing and controlling, such as VIs whose front panels have several charts.
- In the **Web Publishing Tool** dialog box, place a checkmark in the **Request Control** checkbox to immediately request control of a VI embedded in a browser.
- Use the **net address** parameter of the TCP Create Listener function to specify on which network address to listen instead of listening to all addresses. Wire `0` to the **port** input to dynamically choose an available TCP port the operating system determines is valid for use.
- The Close Reference function replaces the Close LV Object Reference and Automation Close functions.
- The 0x10 flag for the Open VI Reference **options** input prompts the user to find missing subVIs.

New VI Server Properties and Methods

Refer to the list of LabVIEW 7.0 features in the *LabVIEW Help* for a list of new properties and methods. LabVIEW includes the following new VI Server properties and methods:

- The Application and VI classes contain several new VI Server properties and methods you can use to open, close, and request control of remote front panels. Refer to the Programmatic Remote Panel Control-Client VI and the Programmatic Remote Panel Control-Server VI in the `examples\remotepanel` directory for examples of using the remote front panel properties.
- Use the Get Control Value [Variant], Set Control Value [Variant], and Get All Control Values [Variant] methods to get and set control values to or from the LabVIEW variant data type. You do not have to flatten data or specify a type descriptor when you use these methods.
- The Value (Signaling) property sets the value of the control and generates a Value Change event. This property updates the value of an object similar to the Value property. However, the Value (Signaling) property also causes LabVIEW to generate an event as if the user had interactively changed the value of the object. National Instruments recommends you use this property only when you rely on LabVIEW generating an event in response to the programmatic value change.

Changes to Existing VI Server Properties and Methods

LabVIEW includes the following changes to the existing VI Server properties and methods:

- Some LabVIEW control properties, such as the Caption property, return references to LabVIEW controls. In previous versions of LabVIEW, these properties returned a new reference each time LabVIEW called the property. If you did not use the Close LV Object Reference function to close the references explicitly, the references stayed in memory until the VI stopped.

  In LabVIEW 7.0, control properties return the same reference instead of creating a new one each time LabVIEW calls the property, which reduces the amount of memory LabVIEW uses to store control property references. Also, you do not need to close the control property reference explicitly.
- If you use the Value property with a cluster, you can wire the value to the Unbundle By Name function and obtain the value of an element in the cluster.

Documentation Enhancements

LabVIEW includes the following enhancements to the LabVIEW documentation resources:

- Refer to the *Quick Reference Card* and the *LabVIEW Help* for an updated list of keyboard shortcuts.
- Refer to the *LabVIEW Application Builder User Guide* for a list of caveats and recommendations to consider when building a stand-alone application or shared library.
- You can display the **Context Help** window by clicking the **Show Context Help Window** button on the toolbar.
- Resize the **Context Help** window to set its maximum size.
- The **Context Help** window displays a description of each property or method as you move the cursor through the shortcut menus of a Property Node or Invoke Node, respectively.
- Move the cursor over most LabVIEW subpalette icons or over an open area of the subpalette to display a description of the subpalette and link in the **Context Help** window. To create a description for a subpalette and to link a subpalette to an HTML file or a compiled help file in a custom palette view, select **Tools»Advanced»Edit Palette Views**, right-click a subpalette, and select **Edit Submenu Documentation** from the shortcut menu.
- VI and function reference topics in the *LabVIEW Help* contain **Place on the block diagram** and **Find on the Functions palette** buttons. Click the **Place on the block diagram** button to place the object on the

cursor so you can place it on the block diagram. Click the **Find on the Functions palette** button to highlight the object on the **Functions** palette. Topics in the *LabVIEW Help* with step-by-step instructions also contain these buttons for every step that instructs you to place an object on the front panel or block diagram.

- In the connector pane images at the top of VI and function reference topics, you can click an input or output name to scroll to the description in the topic for that input or output.
- If you want to link to a `.chm` file on Windows and link to the uncompiled HTML files on Mac OS and UNIX using the **VI Properties** dialog box or the Control Online Help function, the HTML file must be in the `help\html\help.chm` directory, where `help.chm` is the `.chm` filename to which you link on Windows.
- The *LabVIEW Analysis Concepts* manual contains information about the Point-By-Point VIs. Use this manual to learn about the analysis concepts used by LabVIEW. In LabVIEW 6.1 and earlier, the Point-By-Point VIs were documented in the *Getting Started with Point-By-Point VIs* manual.

Other LabVIEW 7.0 Features and Changes

Application Builder Enhancements

Refer to the *LabVIEW Application Builder User Guide* for information about changes introduced between previous versions of the Application Builder and the current version.

New VIs and Functions

LabVIEW includes the following VIs and functions:

- **(Windows, Mac 9.*x* or earlier, and Linux)** Use the Input Device Control VIs to obtain information about the joystick, keyboard, or mouse connected to a computer. The Input Device Control VIs are available only for Mac OS 9.*x* or earlier. Refer to the `examples\input\InputDemo.llb` for examples of using the Input Device Control VIs.
- Use the Analog Waveform VIs and functions to perform arithmetic and comparison functions on waveforms. Use the Special and Numeric Functions VIs to evaluate common mathematical functions.
- Use the following VIs to perform analysis: Align Waveforms (continuous), Align Waveforms (single shot), Analog to Digital Waveform, Bernoulli Noise, Bernoulli Noise Waveform, Binary MLS, Binomial Noise, Binomial Noise Waveform, Digital to Analog Waveform, Gamma Noise, Gamma Noise Waveform, Inverse f Filter Coeficients, Inverse f Filter, Inverse f Noise Waveform, MLS Sequence Waveform, Poisson Noise, Poisson Noise Waveform,

Resample Waveforms (continuous), Resample Waveforms (single shot), Tones and Noise, Tones and Noise Waveform.

Refer to the `examples\measure\resample_align_xmpl.llb` for examples of using these VIs.

- Use the Open URL in Default Browser VI to display a URL or HTML file in the default Web browser. Use the Open HTML Report in Browser VI to display an HTML report in the default Web browser.

  Refer to the Example HTML Report VI in the `examples\reports\withHTML.llb` for an example of displaying an HTML report in a browser.
- Use the Three Button Dialog VI to display a dialog box that contains a message and three buttons.
- Use the Trim Whitespace VI to remove all white space (spaces, tabs, carriage returns, and linefeeds) from the beginning, end, or both ends of a string.
- Use the SO Set Num Buffers VI to set the number of output buffers associated with a sound operation.
- **(UNIX)** Use the Open System Command Pipe VI to open a pipe to a system shell command and return file descriptors that you can pass to subsequent Pipe VIs.
- Use the Search and Replace Pattern VI to search for a regular expression in a string and replace any matching substrings with another string.
- Use the Error Cluster From Error Code VI to convert an error or warning code to an error cluster.
- Use the Request Deallocation function to deallocate unused memory after the VI that contains the function runs.
- Use the Read From XML File VI to read and parse tags from a LabVIEW XML file. Updates to the XML schema reflect the new measurement data types, such as the time stamp, digital waveform, digital data, and dynamic data types. The schema also includes a more comprehensive description for references. The XML schema `LVXMLSchema.xsd` moved from `labview\help` to `labview\vi.lib\utility`.
- The **# available** output of the Get Semaphore Status VI is new and contains the number of tasks that can acquire the semaphore at the current time. The **size** output contains the maximum number of tasks that can acquire the semaphore at a time.
- The **multibyte encoding** input of the string instance of the Read Key VI is new.

Changes to Existing VIs and Functions

LabVIEW includes the following changes to existing VIs and functions:

- The waiting Queue and Notifier functions—Wait on Notifier, Queue Dequeue, and so on—now return error code 1122 in some situations instead of error code 1. The new error code specifies that the queue or notifier refnum on which the function was waiting has become invalid.
- The **configuration file path** input of the Open Config Data VI is a required input. You must specify the **configuration file path** even if you open a reference to a configuration data object.
- The default of the **write configuration file?** input of the Close Config Data VI is TRUE. In LabVIEW 6.1 and earlier, the default is FALSE.
- The MATLAB script node supports strings and paths. Invoking the MATLAB script server no longer launches a MATLAB window. To clear the contents of a MATLAB script, right-click the script node and select **Clear Script** from the shortcut menu. You can run the same MATLAB script in LabVIEW using the MATLAB script node and in MATLAB. However, the current working directory is different for each application. For example, if you call the MATLAB `save` command from the LabVIEW MATLAB script node and you do not specify where you want to save that file, MATLAB saves the file in a different location than if you call the `save` command from a MATLAB script. If you want to save the file in the same directory regardless of whether you call `save` from LabVIEW or MATLAB, add `cd <dir>` to the beginning of the script, where *<dir>* is the directory where you want to save the file.
- The Scale By Power Of 2 function performs a logical shift when the input data (**x**) is unsigned.
- **(Windows)** The In Port and Out Port VIs can read 32-bit values from system memory. If you use the In Port and Out Port VIs on Windows 2000/NT/XP, you do not need to download the `Accesshw.zip` file.
- The Seconds To Date/Time and Get Date/Time String functions no longer round up to the nearest second. These functions display the number of completed seconds. For example, `3.6` seconds would be represented as `3` because `3` is the number of completed seconds. This behavior is consistent with international time standards.
- To set a Comparison function that has two or more inputs to compare elements or aggregates in arrays and clusters, right-click the function and select **Comparison Mode»Compare Elements** or **Comparison Mode»Compare Aggregates** from the shortcut menu.
- The Mathematics VIs are located on the **Analyze** palette.

Miscellaneous

LabVIEW includes the following miscellaneous changes:

- LabVIEW supports multi-seat licenses. Refer to `ni.com/license` for more information about multi-seat licenses.
- **(Windows)** To modify the current LabVIEW installation or to uninstall LabVIEW, select **NI Software** from the Add/Remove Programs applet in the Control Panel. When you modify the installation, a list of NI software appears. Select a product in the list to add or remove individual components or to uninstall the product. You can select multiple products and click the **Uninstall** button to remove all the products you selected.
- When you right-click a path control and select **Browse Options** from the shortcut menu, use the **Pattern Label** text box to enter a label for a custom file pattern in the **Match Pattern** text box. The string you enter in the **Pattern Label** text box appears in the file dialog box next to the custom pattern. Use the Browse Dialog Options:Pattern Label property to set this label programmatically.
- Use the **pattern label** input of the File Dialog function to specify a label for a custom file pattern wired to the **pattern** input. The string you wire to the **pattern label** input appears in the file dialog box next to the custom pattern. Use the **button label** input to specify a label for the **OK** button in the file dialog box. If **select mode** allows the user to select directories, use the **button label** input to specify a label for the **Select Cur Dir** button in the file dialog box.
- When you right-click a path control and select **Browse Options** from the shortcut menu, you can select **new or existing file or directory (use LLBs)** from the **Selection Mode** pull-down menu. Select this option so the user can select a new or existing file or directory including the contents of a `.llb` file from the file dialog box. You also can create a control or constant from the **select mode** input of the File Dialog function and select the **new or existing file or directory (use LLBs)** mode.
- Right-click a path constant and select **Browse for Path** from the shortcut menu to navigate to and select a path.
- In LabVIEW 7.0, FFT calculations continue to increase at a rate roughly proportional to NLogN, and the FFT calculation speed no longer depends on the factorability of size.
- If you configure Perforce as your source code control provider, the SCC user interface changes to resemble the Perforce user interface.
- When you launch LabVIEW or a stand-alone application from the command line, you can pass user-defined arguments. In the command line, enter two hyphens (`--`) surrounded by spaces before the set of user-defined arguments. LabVIEW does not use any arguments after

the two hyphens to launch `labview.exe`. LabVIEW passes the arguments after the two hyphens to the block diagram of the VI you launch. In the VI you launch, use the Application:Command Line Arguments property to read the user-defined command-line arguments passed when the executable launches.

If you use this property in a stand-alone application, you can pass all arguments as user-defined arguments so you do not need to enter the two hyphens before user-defined arguments in the command line. To do so, place a checkmark in the **Pass all command line arguments to application** checkbox in the **Application Settings** page of the **Build Application or Shared Library (DLL)** dialog box.

Refer to the Handling Command Line VI in the `examples\viserver\cmdline.llb` for an example of using the Application:Command Line Arguments property to handle user-defined command-line arguments.

- **(UNIX)** The **Import Clipboard** and **Export Clipboard** items in the **Edit** menu no longer exist. The copy and paste functions work similarly as Windows and Mac OS.
- The **Deallocate memory as soon as possible** checkbox and the **Use default timer** checkbox no longer appear on the **Performance and Disk** page of the **Options** dialog box.