LabVIEW™ Application Builder

Version 5.1

The LabVIEW Application Builder is an add-on package you can use to create executable programs with LabVIEW. Additionally, you can distribute these executable programs without the LabVIEW development software. Consult the *LabVIEW Software License Agreement* for the licensing requirements for distributing executables.

These release notes contain installation instructions, and describe the system requirements for applications created with this version of the Application Builder. You must use the Application Builder 5.1 with the LabVIEW 5.1 Development System.

Contents

321809B-01 *February 1999*

Required System Configuration

Applications you create with the Application Builder Libraries have the same approximate requirements as the development system. Memory requirements depend on the size of your application. Typically, applications require about the same amount of memory it takes to run your VIs in the development system.

LabVIEW applications use a directory for storing temporary files. Some of the temporary files are large and it is best if several megabytes of disk space are available for this temporary directory. You can view or change the temporary directory by selecting **Edit»Preferences...**.

(Sun) You can use a TMPFS file system for this directory to improve performance. Solaris 2 uses TMPFS by default.

If your application aborts unexpectedly, it might leave files behind in the temporary directory. It is a good practice to remove old files occasionally to avoid using up your disk space.

UNIX

LabVIEW applications require an X Windows System server, such as OpenWindows 3, HP-VUE, or X11R6. These applications do not require a specific graphical user interface (GUI) such as Motif or OpenLook, because the program uses `Xlib` to create its own GUI.

LabVIEW Application Builder Libraries for Sun runs on Solaris 2.4 or later. LabVIEW for HP-UX runs on Hewlett-Packard Model 9000 Series 700 computers with HP-UX 10.20 or later. LabVIEW for Linux runs on Linux/x86 computers with RedHat Linux 5.0 or later. LabVIEW for Concurrent PowerMAX runs on PowerMAX version 4.2 or later.

It is best if the workstation has 32 MB of RAM, with 32 MB or more of swap space storage. It is possible for the Application Builder to run on less than 24 MB of RAM, but performance suffers.

Operating System Patches on Concurrent PowerMAX

You must install the following patches, available from Concurrent, so that basic LabVIEW networking functions correctly on Concurrent PowerMAX 4.2:

- inet-005
- One of the following, dependent on your system:
  - base-007 for Power Hawk 610, Power Hawk 620, and all single-processor PowerStack systems
  - base-008 for Power Hawk 640 and multi-processor PowerStack II systems
  - base-009 for Night Hawk systems

Installing the LabVIEW Application Builder Libraries

If you are upgrading from an older version of the Application Builder, install the 5.1 libraries over your old ones.

Windows

The Windows directory on the LabVIEW Application Builder CD-ROM contains libraries for 32-bit Windows (Windows 95/98 and NT) systems. LabVIEW 5.1 is not available for Windows 3.*x*. LabVIEW 5.0.1 has the latest upgrades supported on Windows 3.*x*.

1. Insert the Application Builder CD-ROM.
2. Open the `Windows` directory.
3. Run `setup.exe`.
4. Change the path, if necessary, to point to your LabVIEW directory, then choose **Install**.

☞ **Note** ***Some virus detection programs interfere with the installer program. Check the distribution media for viruses before you begin installation. Then, turn off the automatic virus checker and run the installer. After installation, check your hard disk for viruses again and turn on the virus checker.***

Macintosh

The Macintosh directory on the LabVIEW Application Builder CD-ROM contains libraries for the Power Macintosh. LabVIEW 5.1 is not available for 680*x*0-based Macintosh systems. Use LabVIEW 5.0.*x* to get the latest upgrades supported by Macintosh for the Motorola 680*x*0.

1. Insert the Application Builder CD-ROM.
2. Open the `Mac` folder and double-click the LabVIEW AppLibs Installer icon.
3. After you select the **Install** button, you are prompted to select a destination folder. Select your LabVIEW folder.

☞ **Note** ***Some virus detection programs interfere with the installer program. Check the distribution media for viruses before you begin installation. Then, turn off the automatic virus checker and run the installer. After installation, check your hard disk for viruses again and turn on the virus checker.***

UNIX

The following sections describe how to install the LabVIEW Application Builder for UNIX. Root privileges are not necessary to install these libraries, but you must be able to write to the LabVIEW directory where you plan to install these libraries.

Concurrent PowerMAX

1. Insert the tape into your tape drive.
2. Change the directory to your existing LabVIEW directory. You must have write access to the following directory:

   ```
   cd /opt/labview
   ```

3. Type the following command:

   ```
   tar xv
   ```

HP-UX and Linux

1. Mount the CD-ROM.
2. Type the following UNIX command for HP-UX:

   `cd /cdrom/HP-UX`

   where */cdrom* is the directory where you have mounted the CD-ROM.

   Type the following UNIX command for Linux:

   `cd /cdrom/linux`

   where */cdrom* is the directory where you have mounted the CD-ROM.

3. Run the installation program by typing the following command:

   ```
   ./INSTALL
   ```

4. Follow the instructions on your screen.

Sun

1. Mount the CD-ROM.
2. Type the following UNIX command for Solaris 2:

   ```
   volcheck
   cd /cdrom/cdrom0/solaris2
   ```

   where */cdrom* is the directory where you have mounted the CD-ROM.
3. Run the installation program by typing the following command:

   ```
   ./INSTALL
   ```

4. Follow the instructions on your screen.

Verifying Installation of Application Builder Libraries

Launch LabVIEW after installing the Application Builder Libraries, and choose **Project»Build Application...**. If this item appears grayed out, verify that your `LabVIEW` directory contains an `AppLibs` directory. If this directory is not present, it is possible the libraries were installed in the wrong directory on your computer.

Also, if the libraries are installed correctly, the `examples` directory contains an `appbuild.llb` example. This example is used as part of a tutorial later in this document explaining how to build an application. Refer to the *Application Building Example* section of this document for more information.

Changes to the Application Builder Libraries

Changes Introduced between Version 5.0 and 5.1

This section describes the features that were changed between versions 5.0 and 5.1.

In LabVIEW 5.1, the process for building an application has been streamlined. Previously, to do so you had to save your VIs to a library, then build an application using the **Build Application** dialog box. Further, to build an installer in Windows you had to use the **Create Distribution Kit** dialog box.

Now, in LabVIEW 5.1, you can use the **Build Application** dialog box to do all of these operations. You can configure the application to various settings within the tabs on the **Build Application** dialog box. After you define these settings, you can save them in a script so that you can easily rebuild the application. For complete details, refer to the *How to Build an Application* section, later in this document.

Changes Introduced between Versions 4.1 and 5.0

The following list contains the features that were changed between versions 4.1 and 5.0:

- When building an application, the method to select whether the application **File** menu has **Open** and/or **Exit** menu items has been changed. You now use the new run-time menus feature to alter the menus. For more information, see the section *Customizing the Menubar* in Chapter 6, *Setting Up VIs and SubVIs*, in the *G Programming Reference Manual*.
- **(Windows 95/NT)** When building an application, you now can choose to enable ActiveX server support. For more information, see the *Customizing Application Features* section in this document.

Changes Introduced between Versions 4.0 and 4.1

The following bug was fixed between versions 4.0 and 4.1:

- If the installed libraries are read-only, building an application fails.

LabVIEW Application Builder Libraries Version 4.0.1 corrects the following problems:

- Run-time pop-up options were not available in a built application.
- **(Windows)** The menu option for data logging was missing in a built application.
- Using the keyboard to select components of clusters did not work in a built application.
- Editing the cursor colors caused the application to crash in a built application.
- The **Autoscale** option in the pop-up menu for a graph did not work correctly in a built application.
- In a built application, the Help description for a control was not displayed if the control had a keyboard shortcut.

Features of LabVIEW Applications

For more information about LabVIEW features refer to Chapter 27, *Managing Your Applications*, in the *G Programming Reference Manual*. This chapter contains tips for managing the source files of multiple developers and describes how to use the VI History item.

Standard Features

LabVIEW applications feature a simplified user interface that permits only the operation of VIs. The menus do not contain editing options. For example, the **Save** command and the **Functions** and **Controls** palettes are not present.

Menus display items related to VI operation. Because you cannot edit the VI, pop-up menus are short—displaying the same items the development system displays when a VI is running.

(Windows and UNIX) Users access a pop-up menu by clicking a control or indicator with the right mouse button.

(Macintosh) Users access a pop-up menu by holding down the command key and clicking a control or indicator.

The capabilities available to the user include the following:

- Operate controls and change their values.
- Interact with strip chart and graph indicators.
- Change the scale limits.
- Set controls, indicators, and array elements to default values.
- Use the pop-up menu of a control or indicator to cut, copy, or paste data from a control or indicator to another control.
- Use the pop-up menu of a control or indicator to view the description of the item, and perform additional run-time operations, such as showing the control palette of the graph.
- Use any execution palette button the developer has not disabled.
- Log and print the front panel.
- View the **Show VI Info...** information for a VI.
- Use the Help window to see descriptions of controls and indicators.

How to Build an Application

Complete the following instructions to build an application in LabVIEW.

Building an Application

Complete the following instructions to build an application in LabVIEW.

1. Select **Project»Build Application…**. The **Build Application** dialog box appears. The **Build Application** dialog box contains the following tabs: **Target**, **Source Files**, and **VI Settings**. On some platforms you also have the **App Settings** and **Installer** tabs. You can create a new build or load a build file that you created previously.
   - If you want to create a new application, proceed to Step 2.
   - If you already have a build file, click **Load…** and choose the `.bld` file to load. Then proceed to Step 6.
2. From the **Target** tab, specify the following information:
   - **Application name**—The name of the application you are creating.
   - **Destination directory**—The path and name of the directory in which to create and save your new application.
   - **Support file directory**—The path and name of the directory in which to save any support files.
3. Under the **Build** section of the **Target** tab, choose one of the following two options:
   - **Single application containing all VIs**—Check this option to create a single application containing all of your VIs.
   - **Small application with external file for subVIs**—Check this option if you want to keep the main application small.
4. From the **Source Files** tab, you can configure the VIs that make up your application. You can specify top-level VIs, dynamically loaded VIs, and additional non-VI files (such as readme files). You can update the file list automatically as VIs are added or removed from your hierarchies. Depending on what kind of source files you want to add, complete the instructions below.
   a. If you want to add top-level VIs, click **Add Top Level VI…**. The **Open** dialog box appears where you can enter a file name, or browse to find the VIs you want to add. When you select a top-level VI, LabVIEW automatically includes all its subVIs and related files (such as menu files or DLLs). Top-level VIs are automatically opened and run when you start an application.

b. If you want to add dynamic VIs, click **Add Dynamic VI…**. If your VI dynamically calls any subVIs using the VI Server, LabVIEW cannot detect them automatically, so you must add them by using this option.

c. If you want to add support files, click **Add Support File…**. When you use this option, data files copy over to your application directory automatically. In addition to VI files (VIs, controls, menus, external subroutines, and so on), you can determine the set of DLLs referenced by your application. Because there are DLLs you might not want to redistribute, LabVIEW includes only those DLLs that are within the source hierarchy directories and the LabVIEW directory. If you want to include DLLs that are in the System directory, for example, you can include them as additional non-VI files manually.

d. If you want to remove a file from the list, click the file to highlight it and click **Remove File**.

5. Click **Save** to save the information you have entered. The **Save As** dialog box appears. Enter a file name with a `*.bld` extension to save the information you have entered into this dialog box.
6. Click **Build**. The **Build Status** dialog box appears.
7. After the build operation finishes, click **Done** to close the **Build Application** dialog box.

Customizing Application Features

1. If you want to customize some destination settings, select **Custom Destinations…** from the **Source Files** tab. The **Destination Settings** dialog appears, in which you can configure the following settings:
   - You can modify your destination directory. You might use this option to place a support file in a specific subdirectory.
   - **(Windows)** If you want to add a program item to your **Start** menu as part of an installer, select the **Create program item** checkbox and enter the name.
   - **(Windows)** If you are creating an installer, you can specify how you want to **Replace existing files**. Select **Never**, **Ask**, **If Newer**, or **Always**, depending on how or if you want to be prompted.
2. **(Windows** and **Macintosh)** From the **App Settings** tab, you can customize the features in your application. You can choose to specify the memory size for the Macintosh, or customize icons and ActiveX server features on Windows.
   a. **(Windows)** If you want to specify your own icon, click the **Custom icon** checkbox and designate the path to the icon (`.ico` file).

b. **(Windows)** If you want to enable the ActiveX server, click the **Enable ActiveX server** checkbox. Your application can then respond to requests from ActiveX clients. The functionality of the ActiveX server in your application is a subset of the LabVIEW ActiveX server. When you build an application `myapp.exe`, an ActiveX type library `myapp.tlb` is also created along with the executable. The type library defines a createable class, *Application*, and a dispatch class, *Virtual Instrument*, and exports the properties and methods for these classes. You can find the Help for these properties and methods in `lvcomm.hlp` in the LabVIEW `Help` directory. When you distribute the application make sure the type library and the help file are located with the executable.

   When you assign the name of the application to the server name, your application is uniquely identified in the system registry. Once you build the application, you should run it at least once to enable registry with the system. After the application is registered, ActiveX clients access the server objects using server names. For example, if you specify the server name as `myapp`, clients instantiate an application object using `myapp.application`.

c. **(Macintosh)** Use the Memory Size control to specify the memory allocated to the application.

Modifying VIs as Part of the Build

Use the **VI Settings** tab to specify the modifications to your VIs that you want to make part of the application build. You can choose to enable or disable various window option and execution option VI Setup settings. These settings apply to the build process only and do not affect your original source VIs.

LabVIEW removes debugging code, block diagrams, and unnecessary front panels, making your application as small as it can be. The removal of front panels is a new feature with LabVIEW 5.1. LabVIEW can detect which panels are necessary in almost all cases. However, if you open a front panel dynamically using the VI Server, you must specify that the panel is needed using the **VI Settings** tab.

You can edit only a single row at a time. By default, all unnecessary panels are removed. You can override the default and include the panel by setting the **Remove Panel** option to **No**.

To change settings, select a VI so that it is highlighted in the list. Click **Edit Build Settings…**. The **Edit Build Settings** dialog box appears. For each setting you can choose **yes**, **no**, or **no change**. When you have made all the settings, click **Change**. Verify that all the settings are the way you want them for each VI in the build.

☞ **Note** ***This completes the build application process on the UNIX and Macintosh platforms. The steps described in the following section apply to Windows only.***

Creating an Installer (Windows only)

1. From the **Installer** tab, click the **Create installer** checkbox. Verify the following sections of this tabbed page. If you create an installer, the installer is written to the directory that contains your application. The disk images are created in a `disks` subdirectory of the destination directory that you specified on the **Target** tab. This directory will contain a setup program as well as files named `data.001`, `data.002`, and so on. If you plan to put the disk images on floppy, it is best to copy the `setup` and `data.001` files to the first floppy, copy the `data.002` file to the second floppy, and so on.
   - Installation name
   - Start menu program group
   - Default installation directory
   - Installation language
   - Media size
   - Extra space on first disk (KB)

   The **Media size** item lets you specify how the file is to be segmented—for 720 KB, 1.2 MB, or 1.4 MB floppies. Even if you plan to distribute the files by CD, it is necessary to segment them. However, if you want to run the installer from a CD or from your drive, you can place all of the files in the same directory and run the setup program from that directory.

   The **Extra space on first disk (KB)** item lets you reserve space on the first disk. You might reserve space on the first disk if you want to put a readme file on the first floppy.
2. Click the **Advanced** button. The **Advanced Installer Settings** dialog appears.
   a. If you would like to create an uninstaller, click the **Create uninstaller** checkbox.
   b. If you would like to run a program after the installation, click the **Run executable after installation** checkbox and enter the executable and command line argument information.
3. Select the **Run executable after installation** item if you want to run a program after the installation completes. Additionally, you can use this item to run a program that finishes the installation. For example, you might write a DOS batch program or a C program that modifies a `.ini` file or a `registry` file. Install the file as part of your installation and then run it afterwards to make the necessary modifications. The file that you run must be one of the files that you install.

4. If you choose to run an executable after the installation completes, you can use the **Command line arguments** to specify arguments passed to the program. In addition to specifying standard arguments, you can embed any of the following items in the command line argument string:

   `%dest` The application installation directory chosen by the user

   `%src` The directory that contains `setup.exe`

   `%group` The installation program group name

   `%name` The installation name

   If any of these options are present at installation time, they are replaced with the proper values before the arguments are passed to the executable.

Application Building Example

Complete the following steps to explore application building using your LabVIEW development system:

1. Open the Sample VI, located in `examples\appbuild.llb`. This VI calls some Analysis VIs, including the Histogram VI, which call external subroutines. Items in the **VI Setup…** dialog box are currently configured to hide a number of the attributes of the window.
2. Run the Sample VI to see its behavior. When you are finished, click the **STOP** button. Do not click the **QUIT** button unless you want to quit LabVIEW.
3. Close the Sample VI.
4. Open the **Build Application** dialog.
5. Name the application (**Windows**) `sample.exe` or (**Macintosh** and **UNIX**) `sample`.
6. Change the destination directory to a reasonable destination.
7. From the **Source Files** page, add `examples\appbuild.llb\sample` as a top-level VI.
8. Add `examples\appbuild.llb\About Sample` as a dynamic VI. The application will automatically use any VI whose name begins with About as an **About** dialog box.
9. Build the application.
10. When you run the application, the top-level VI(s) are automatically opened and run. Also, the **STOP** button in the toolbar is automatically turned off. All subVIs of the sample VI were automatically added to the application.

11. At this point, you might consider using some of the additional options in the **Build Application** dialog to simplify creating your application. For example, you might want to hide the toolbar in your application, while keeping it visible in your source VIs for easier debugging. You can use the **VI Settings** page to create transformations such as this. In addition, on Windows you can use the **App Settings** page to specify a custom icon or the **Installer** page to create an installer.
12. By default, applications use simplified menus. Look at the menu of the sample application to see the contents. You might want to customize this menu further to remove options such as **File»Open** or the datalogging options in the **Operate** menu. You can use the menu editor (**Edit»Edit menu**) to specify a custom menu for your VIs. Also, you might want to change the **Exit** option. Rather than allowing the application to handle it, make it a custom option so that you can do your own processing (such as closing files and terminating I/O) before calling the Quit Application function.

Distributing Your Applications

The following sections describe some relevant issues concerning the distribution of LabVIEW applications.

Additional Files Required by Applications

It might be necessary to distribute additional files with your application. If required for execution or other operations, these files must be placed in the same directory as the application.

If your application uses a GPIB or data acquisition board, the user must install the hardware drivers that come with their boards.

(**Windows** and **UNIX**) If your application uses the serial port, you need to include the `serpdrv` file from your `LabVIEW` directory with your application. In addition, on Windows you need to include the `daqdrv` file if you use data acquisition. In the `Build Application` directory, you can include these files by using the **Support file** option.

(Windows) When you develop an executable program with LabVIEW on Windows and ship it to another computer, you must also include the LabVIEW Run-Time Engine. The computer on which the program runs must install this component using the LabVIEW Run-Time Engine Installer before the program executes. If you distribute a program using Build Application, the Run-Time Engine is installed automatically. This enhancement greatly reduces the size of the executable program.

☞ **Note** ***After the Run-Time Engine is properly installed on a machine, it can run any executable program developed with that version of LabVIEW. Only with the first program sent to each computer is it necessary to include the Run-Time Engine.***

(Windows) The following items are automatically installed by the LabVIEW Run-Time Engine:

- If you are using an ActiveX type library or ActiveX control in your application, the type library and help file must be included. Also, the library will need to be registered. The best way to handle this is with separate installers. Many ActiveX libraries and controls have licensing issues you should investigate before redistributing ActiveX components. If you want to integrate the ActiveX installation with your application installation, use a third-party installer, such as InstallShield or WISE.
- The ActiveX Container uses a DLL named `ole_lv5container.dll`, which is located in the `resource` directory. If you build an application that includes ActiveX controls and move it to another machine, you must install this file in the same directory as the built application or in the `System` directory.
- LabVIEW requires two system DLLs that might not be installed on all systems. These are `msvcrt` and `mfc42.dll`. The file `msvcrt.dll` is always required and `mfc42.dll` is required only if the application uses the ActiveX container object. If they are not already present, these files are placed in your Windows 95 `system` directory (`system32` directory on the NT platform) by the LabVIEW installer. You must make sure these files exist on the computer which runs the application. They can be placed in either the Windows 95 `system` directory (`system32` directory on the NT platform) or in the same directory as the application. If you put them in the Windows directory take care not to overwrite newer versions of these DLLs that might have been installed by other applications.

(Macintosh) If you are building an application for the Power Macintosh that uses analysis routines or Program to Program Communication (PPC) VIs, put a copy of the `Shared Libraries` folder in the folder that contains your application. You can use the **Support files** option to add these to your distribution.

Distribution Rights

Refer to the *LabVIEW Software License Agreement* in your software package for information on the distribution rights for your platform.

Additional Notes

The following sections contain some additional information that might be useful in setting up your applications.

Setting Preferences

Your application has a **Preferences** dialog box. If you make changes, preference information for your application are written to **(Windows and Macintosh)** a preferences file or **(UNIX)** the `.labviewrc` file. The format for this information is the same as for LabVIEW; for more information see Chapter 7, *Customizing Your Environment*, in the *G Programming Reference Manual*.

(Windows) The only difference is the file name, which is the application name instead of `labview`. For example, if your application is named `simple.exe`, the preferences are stored in `simple.ini`. Remember to include your preference file (`.ini`) with your application. You can do this by treating it as a support file in the **Build Application** dialog.

(Macintosh) The only difference is the file name, which is the application name instead of `labview`. For example, if your application is named `Simple`, the preferences are stored in `Simple Preferences`. Remember to include your preference file with your application. You can do this by treating it as a support file in the **Build Application** dialog.

(UNIX) To create a preferences file for your stand-alone application, you must create a `.labviewrc` file on the target machine and place it inside your home directory. Inside this file you create preferences using the following format: `<application file name>.<preference>`. You can do this by treating it as a support file in the **Build Application** dialog. For example, the `.labviewrc` file for LabVIEW has its preference set as `labview.<preference>`. An easy way to create a new executable specific preference is to set your preferences in LabVIEW and then place a copy of the `.labviewrc` file in your home directory of the target computer. All the user has to do at this point is change all the `labview` prefixes to the executable file name. The preferences file name, however, is still called `.labviewrc`.

Using the VI Setup... Option to Limit VI Options

When you design an application, consider which user options you might want. For example, with the LabVIEW Development System, it is convenient to have an **Abort** button so users can easily test and halt VIs. Because this button aborts the program immediately—sometimes in the

middle of input and output of sensitive data—you probably do not want the user to halt the VI by this method. Instead, use a front panel control to stop the program synchronously.

You can use the **VI Setup** item in your VIs to make execution operations—such as the **Abort** button—available to the user. If the VI is to be used as a subVI, you can use the **SubVI Node Setup...** command to specialize operation of the subVI, such as configuring its front panel to open when the subVI is called.

Many of these options are also available in the **VI Settings** page of the **Build Application** dialog. You can use this page to specify changes that should be applied as part of the build. This allows you to configure VIs in a convenient manner for debugging, but change them in your resulting application. By default, the Abort button is removed as part of the build process.

You can disable the run-time pop-up menu for your VIs, so that users cannot set controls to default values or turn on autoscaling in graphs.

Additionally, you can customize your panels. For example, you can hide scrollbars or make specific VIs function like dialog boxes. When you choose **VI Setup...»Dialog Box**, the panel is modal, which means the user cannot interact with other panels while the panel is active.

Providing Help Information

As a VI developer, it is best that you document your VIs for other users, including all information necessary to load and operate the VIs. The regular LabVIEW documentation set is copyrighted material. Do not ship this documentation with the applications you create.

To create online help, you might want to enter information in the description field of the **Show VI Info...** dialog box for each panel. You might also want to place information in the **Description** dialog box for controls and indicators. When the user opens the Help window and moves the cursor over indicators, the Help window displays description information.

Common Errors

UNIX

The following table lists common errors that might occur when you launch LabVIEW for UNIX.

Table 1. Common LabVIEW Launch Errors on UNIX

Error Message/Description	Error Message/Description
"Xlib: connection to :0.0 refused by server"	**Probable Cause**—Trying to run LabVIEW as a user who does not have permission to open a window on the display server. Typically seen after running the `su` command to temporarily become a different user, such as root (superuser). **Solution**—Exit the `su` command and launch LabVIEW as the login user.
"Client is not authorized to connect to server"	Internal error during connection authorization check.
"Executable version doesn't match resource file"	**Probable Cause**—Version of LabVIEW executable does not match version of `labview.rsc`. **Solution**—Verify that the `appResFilePath` parameter in the configuration file correctly sets the path to the `labview.rsc` file.

321809B-01 Feb99