NATIONAL INSTRUMENTS™
The Software is the Instrument®

Release Notes

LabVIEW® Application Builder for UNIX

Version 4.1

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 4.1 with the LabVIEW Development System 4.1.If you are upgrading from an older version of the Application Builder, install the 4.1 libraries over your old ones.

Contents

Required System Configuration

Applications that you create with the Application Builder Libraries have approximately the same 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 for Sun and HP-UX ships on CD only. LabVIEW for Concurrent PowerMAX ships on 4 mm DAT Tape. LabVIEW applications require an X Window 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 come in two versions, one for Solaris 1 and one for Solaris 2. LabVIEW applications for HP-UX run on Hewlett-Packard Model 9000 Series 700 computers under HP-UX 9.0.3 or later, and HP-UX 10.0 or later.

The workstation should have 32 MB of RAM, with 32 MB or more of swap space storage. The Application Builder can run on less than 24 MB of RAM, but performance suffers.

LabVIEW applications use a directory for storing temporary files. Some of the temporary files are large, so we recommend that you have several megabytes of disk space available for this temporary directory.

The default temporary directory is `/tmp`. You can change the temporary directory by selecting **Edit»Preferences...**.

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

(Sun) You can use a TMPFS file system for this directory to improve performance. For Solaris 1.*x*, refer to the Sun *System and Network Administration* manual, part number 800-3805-10, for more information about the TMPFS file system. Solaris 2 uses TMPFS by default.

Operating System Patches on the Sun

LabVIEW applications require the same patches to the operating system as the LabVIEW development system.

When you build an application for Sun OS 4.1.3, customers who use it must obtain the latest revision of the following patch from Sun:

100458-*xx*: Setitimer sometimes fails to deliver SIGALRM

If you plan to run LabVIEW applications under Solaris 2.3, you should obtain the latest versions of the following patches from Sun:

101318-*xx*: jumbo patch for kernel
101347-*xx*: fixes for ttcompat
101409-*xx*: jumbo linker patch
101489-*xx*: libthread jumbo patch

There are no other patches necessary to run LabVIEW under Solaris 2.4 or later.

Installing the LabVIEW Application Builder Libraries

The following section describes how to install the LabVIEW Application Builder for UNIX. You do not need root privileges to install these libraries, but you must be able to write to the LabVIEW directory where you plan to install these libraries.

Installation Procedure for 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 this directory.
   `cd /opt/labview`
3. Type the following command:
   `tar xv`

Installation Procedure for Sun and HP-UX

1. Insert the first floppy disk into the floppy disk drive.
2. Type the following UNIX command for HP-UX (the device name `c20Ad1s0` might be different on your machine):
   `tar xvf /dev/rfloppy/c20Ad1s0 INSTALL`

   Type the following UNIX commands for Solaris 1:
   `tar xvf /dev/rfd0c INSTALL`
   Type the following UNIX command for Solaris 2:
   `volcheck`
   `tar xvf /vol/dev/aliases/floppy0 INSTALL`
3. Run the installation program by typing the following command:
   `./INSTALL`
4. Follow the instructions on your screen. You will be prompted to insert the subsequent floppy disks.

What the Application Builder Libraries Add to LabVIEW

If you launch LabVIEW after installing the Application Builder Libraries, choose **Project»Build Application...**. If this option appears grayed out, verify that your `LabVIEW` directory contains an `AppLibs` directory. If this directory is not present, you might have installed the libraries into the wrong directory on your computer.

In addition, the `examples` directory should contain 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 4.0 and 4.1

The following list contains features that were added or changed, or bugs that were fixed between versions 4.0 and 4.1:

- If the installed libraries were read-only, building an application would fail.

Changes Introduced between Version 3.1 and 4.0

The LabVIEW 4.0 Application Builder Libraries have been upgraded as a part of a major upgrade to LabVIEW itself and contain no significant new features:

Changes Introduced between Version 3.0.1 and 3.1

The following features were added or changed between versions 3.0.1 and 3.1:

- The Application Builder Libraries now include the functionality of the run-time system. When you build an application, you can choose if you want to embed a library of VIs within the application. For more information on this feature, read the *Customizable Features* section in these release notes.
- When you build an application, you can select whether the application **File** menu has an **Open** menu option and a **Quit** menu option. If you remove the **Quit** option, your VIs should use the Quit LabVIEW function to end the application.
- LabVIEW applications can now call VIs outside of themselves. When you embed VIs, you do not have to embed every VI. Also, you can use the VI Control VIs to dynamically call VIs that are not embedded in the application.
- LabVIEW uses memory in the same way as other applications. As a result, you no longer need to specify the `appTotalMem` or `totalMemSize` preferences.

Features of LabVIEW Applications

For more information about LabVIEW application features refer to Chapter 24, *Managing Your Applications*, in the *LabVIEW User Manual*. That chapter contains tips for managing the source files of multiple developers and describes how to use the VI History option.

Standard Features

LabVIEW applications feature a simplified user interface that allows only the operation of VIs. The menus do not contain editing options. For example, the **Save** option and the **Functions** and **Controls** menus are not present and cannot modify your VIs or view the diagrams.

Menus display options related to VI operation. Because you cannot edit the VI, pop-up menus are short—displaying the same options the development system displays when a VI is running. Users access a pop-up menu by clicking on a control or indicator with the right mouse button.

The options 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 that 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.

Customizable Features

When you build a run-time application, you can customize the following options:

- Do you want to embed a VI library in the application?

  If you choose to embed a VI library in the application, the library and a LabVIEW run-time engine become a single file. When you launch the file, it automatically opens all top-level VIs in the library. If you do not embed a VI library, when you launch the application you can use it to open any VI, assuming the VI was saved with a development system for that platform.

  By embedding a VI library, you can create a complete stand-alone application, one that prevents the user, or customer, from accessing the source VIs—even if the user has the development system. The advantage to not embedding a VI library is that you can use the same run-time engine for multiple sets of VIs. This reduces the disk space usage for a customer who needs to run multiple VIs.

  Additionally, you can use a combination of these two solutions. If you embed VIs within a library, they can still call subVIs that are outside of the application. You might want to do this when you have a set of VIs common to two applications, a set of VIs that you might need to upgrade after the user receives the application, or a large number of VIs to call (to keep the base size of the application down). One disadvantage of not embedding every VI is that the subVIs can be used in another development system, because the users can view the diagrams.

- Do you want the application to have an **Open** menu option?

  If you enable the **Open** menu option, the application can open and run any VI in the file system, regardless of whether the application has an embedded VI library.

  If you plan to ship multiple VI applications to a customer, you may want to have a run-time application with an **Open** menu option. This way, you can send a single run-time application to the customer, which he can then use to open and run your VIs.

- Do you want the application to have a **Quit** menu option?

  You may want to remove the **Quit** menu option if you want to control when the user can quit. For example, you may want to prevent the user from quitting during I/O, or you may want to clean things up before you allow him to quit. If you remove the **Quit** option, you must provide another method for the user to quit

your application. Your application can call the Quit LabVIEW function when you want the application to quit.

- Do you want a customized **About...** dialog box?

  When you build an application, you can supply an About VI that runs when the user selects **Help**»**About…** from the **Help** menu. When you do not supply and About VI, the application has a basic, default About dialog box. Notice that if you want an About dialog box, it must be part of an embedded VI library. If you embed a VI library, at least one VI within it (excluding the About VI) must be marked as Top Level.

How to Build an Application

This section describes how to build an application. First, you will probably want your top-level VI(s) to run when opened. You can turn this feature on by selecting **Run When Opened** from your top-level VI(s) VI Setup dialog. If you want all of your VIs embedded within the application, save your application VIs into a single VI library by selecting **Save with Options**. You can also save a VI in this library to use as an About VI, so users can view information about your application (such as the full name, version number, company name, copyright information, and so on).

Next, choose **Project**»**Build Application...** option to build the application. If you choose to embed a VI Library, and no VIs are marked as Top-Level, the **Build Application...** option brings up the **Edit VI Library** dialog. The VI(s) you mark with the Top-Level option open when the application is launched. The following paragraphs describe these steps in greater detail.

Save the VIs for the Application, If Embedding

Choose **File**»**Save with Options...** to save a hierarchy of VIs for an application. If you click on the **Application Distribution** option, your program prompts you to select the VI library or directory where you want to save the hierarchy. Enter the name of a new library that you want to use to build the application. This selection automatically saves the VIs without their diagrams and includes any external subroutines referenced by the VIs in the VI library.

If you want the top-level VI(s) to run every time the application is launched, select **Run When Opened** in the VI Setup dialog box of the VI(s).

Large applications require additional time to save the VIs into the library.

Create and Save an About VI (Optional)

Most applications have an About dialog box that displays information about the application and the user or company that designed it. You can create an About VI that LabVIEW executes when a user selects **Help**»**About…**. You can have only one About VI per application, and you can only have one if you embed a VI Library. If you do not supply an About VI, LabVIEW displays a default dialog box like the one displayed in the LabVIEW development system.

To create your own About VI, create a VI and save it so that its name begins with the word *About* (the first letter must be capitalized, with the subsequent letters in lowercase). When the application is launched it looks for a VI beginning with the word *About*.

If the user selects the **About…** menu option and you have installed an About VI, the VI will run. When it finishes execution, LabVIEW closes it automatically.

The About VI you create can share subVIs with your application VIs. However, your About VI cannot be a subVI in an application VI because the About VI cannot run while an application VI is running.

☞ **Note:** ***Your About VI must contain a message indicating that your application was created using LabVIEW from National Instruments. Please read the*** *Distribution Rights* ***section in the*** *LabVIEW Software License Agreement* ***for the copyright notice you must use to legally distribute your applications.***

Use the Build Application Option

Select the **Project**»**Build Application...** to create an executable.

If you want to embed a VI library, click on the **embed** option. A dialog box appears prompting you to select which VI library you want to use to build an application. Select the VI library you created in the first step of this section.

In addition, you can choose whether you want an **Open** menu option and a **Quit** menu option.

When you finish making selections, click **OK**. If you embed a VI library and no VIs in the library are marked with the **Top Level** option, LabVIEW displays the **Edit VI Library...** dialog box so you can select which VIs open at launch time. Use the **Top Level** option to mark the VI(s) that you want to open when you launch the application. Most applications consist of a single top-level VI that calls other VIs. However, you can create applications that consist of multiple top-level VIs that open when you launch an application.

After the prompt, enter a name and destination for the application. The build process can take a few minutes for very large applications.

Complete Installation

If your application uses serial port functionality, place the `serpdrv` interface file in the directory that contains your application.

You must install the hardware drivers for any GPIB or data acquisition devices you use with your application.

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. Options in the **VI Setup…** dialog box currently are 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. Examine the About Sample VI, which is also in `examples\appbuild.llb`. This VI serves as the About dialog box for this application. When this VI executes, it acts similarly to a dialog box, in that it prevents you from interacting with other windows while it is open.
4. In the edit mode, select **VI Setup...** in the pop-up menu of the icon pane of the Sample VI. Then select **Run When Opened**»**OK**.
5. Choose **File**»**Save with Options...**»**Application Distribution**»**Save**. When prompted, enter the name `sample.llb` and then click on **Select**.

6. Save a copy of the About Sample VI into `examples\sample.llb` and click on **OK**.
7. Now you can build the application. Select **File»Build Application…**. Click on the **Embed VI Library** button and choose `sample.llb` and click on **OK**. LabVIEW then prompts you to mark which VIs should open when you launch the application. Choose the Sample VI from the displayed list, and select **Top-Level»OK**. A dialog box prompts you for a destination and name for the application you want to build. Move upward in the file hierarchy to the top level of your LabVIEW directory, and name the application `sample`.

☞ **Note:** ***If you do not build the application at the top-level of the LabVIEW directory, you might need to place several files in the same directory as the application. These files communicate with hardware. See the*** *Additional Files Needed by Applications* ***section for a list of the files that you should store with your application.***

8. Quit LabVIEW, and run the `sample` application. It should launch and then automatically open and run the Sample VI. Look at the menu options that are now available. Select **Help»About…**. When you finish, click on the **QUIT** button on the front panel of the application.
9. In practice, you may want to completely remove the **STOP** button from the front panel. If the top-level VI stops, the application does not automatically quit, because that may not be the desired behavior. You should structure most applications so that the **Abort** option is disabled on the top-level VI, and the VI has a **Quit** option that calls the Quit LabVIEW function, located in **Functions»Advanced**.

Distributing Your Applications

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

Additional Files Needed by Applications

You might need to distribute additional files with your application.

If your application uses serial port functionality, include the `serpdrv` file. If your application uses a GPIB or DAQ device, the user must install the hardware drivers that come with their devices.

To keep your original system preferences (for example, the system fonts) when changing systems, include your preference file (`.labviewrc`) with your application. Refer to the *Setting Preferences* section for more information.

Distribution Rights

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

Packaging Your Files for Distribution

A LabVIEW application can be quite large. A core run-time system is smaller than the development version, because it does not require the compiler or the editor. In addition, the run-time system saves the VIs without diagrams, which usually accounts for at least half of the size of your VIs. Even so, most run-time applications will not fit on a single floppy disk. To distribute your applications, you might want to put them on a larger capacity medium, such as a CD or magnetic tape, or you can compress the applications and possibly create an installer for them.

You can use the Unix `tar` command to group the application files into a single file for distribution.

Additional Notes

The following sections contain some additional information that may 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 will be written to the `.labviewrc` file. The format for this information is the same as for LabVIEW; for more information see Chapter 8, *Customizing Your LabVIEW Environment* in the *LabVIEW User Manual*. The only difference is the configuration token prefix, which will be the application name instead of `labview`. Remember to include your preference file with your application.

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

When you design an application, you should consider which user options that you want. For example, with the LabVIEW Development System, it is convenient to have an Abort button so users can easily test and halt VIs. For an application, you probably do not want the user to halt the VI with this button, because it aborts the program immediately—sometimes in the middle of input and output of sensitive data. Instead, use a front panel control to stop the program synchronously.

You can use the **VI Setup** option in your VIs to make execution operations—such as the **Abort** option—available to the user. If the VI will 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.

You may want to disable the following three options: **Abort**, **Close Box**, and **Free Run**. You may want to hide all of the buttons, and then configure the top-level VI to run automatically when the VI is loaded. If you disable the Abort button, verify that your program does not accidentally run in an infinite loop.

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, you should document your VIs for other users, including all information necessary to load and operate the VIs. The regular LabVIEW documentation set is copyrighted material and should not be shipped with the applications you create.

To create online help, you may want to enter information in the description field of the **Show VI Info…** dialog box for each panel. You may 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

The following is a common error that occurs when you launch a LabVIEW application:

Error Message/Description	Probable Cause/Solution
“Xlib: connection to:0.0 refused by server”, OR “client is not authorized to connect to server”, OR “internal error during connection authorization check”	Trying to run the application 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). Exit the su command and launch the application as the login user.