NATIONAL INSTRUMENTS™
The Software is the Instrument®

Release Notes

LabVIEW® Application Builder for Windows

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

320705D-01

May 1997

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.

Installing the LabVIEW Application Builder Libraries

The LabVIEW Application Builder package contains disks for installing the libraries under 16-bit Windows (Windows 3.*x*) and 32-bit Windows (Windows 95 and NT).

Installation Procedure

1. Run `setup.exe` on your Application Builder disk.
2. Change the path, if necessary, to point to your LabVIEW directory. Then choose **Install**.

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.

Also, the **Project** menu should have a **Create Distribution Kit** option. After you build an application, you use this option if you want to create an installer for your application. Additionally, **Create Distribution Kit** automatically splits up your application into files that can fit on floppies.

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.
- The Create Distribution option uses a newer install program. This new version fixes several floppy installation problems and uninstaller problems.

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:

A new **Create Distribution Kit** option in the **Project** menu makes it easy for you to distribute your VIs. Selecting this option, brings up a dialog box that helps you build an installer for your application. See the

Distributing Your Application section in these release notes for details on how to use this new feature.

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 an **Exit...** menu option. If you remove the **Exit...** 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 an **Exit...** menu option?

  You may want to remove the **Exit...** 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 **Exit...** 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 an **Exit...** 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, or data acquisition place the `serpdrv` or `daqdrv` interface files 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.exe`.

☞ **Note:** ***If you do not build the application at the top-level of the LabVIEW directory, you 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.exe` 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 or data acquisition functionality include the `serpdrv`, or `daqdrv` files. 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 (`.ini`) 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

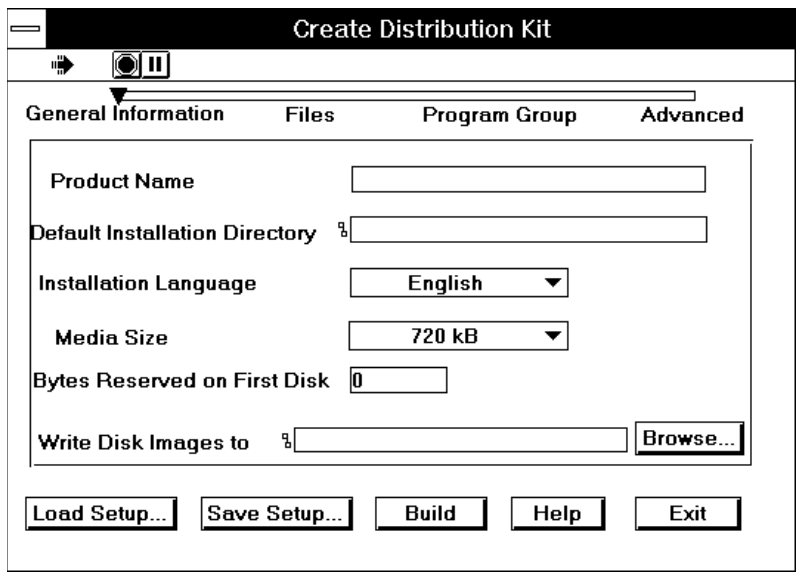
A LabVIEW application can be quite large. A core run-time system is smaller than the development version, because it does not require the compiler or the editor. In addition, the run-time system saves the VIs without diagrams, which usually accounts for at least half of the size of your VIs. Even so, 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.

The Windows version of the Application Builder Libraries adds a **Create Distribution Kit** option to the **Project** menu. This command makes it simple to create professional installers for your LabVIEW applications.

Create Distribution Kit

When you select **Project»Create Distribution Kit** the following dialog box appears:



To create an installer, fill out the information in this dialog and then click on the Build button at the bottom. This compresses the files, builds an installer, and writes the disk images to the path specified in **Write Disk Images**. You can run the installer image from that directory or copy it to floppies or other distribution media.

If you are not sure of the meaning of a field or button, click on the **Help** button at the bottom of the dialog to bring up the Help window in

LabVIEW. You can then put your cursor over any field or button, and a description of the field or button appears in the Help window.

You can use the **Save Setup** option to save the information that you enter in this dialog box. Subsequently, you can use the **Load Setup** option to retrieve information.

The **Create Distribution Kit** option has four dialog boxes: General Information, Files, Program Group, and Advanced. To switch between these dialog boxes, click on the name of the box you want to go to or slide the slider.

General Information

You use the General Information dialog box to describe appearance and default options of the installer, and to specify a location on your drive to write the installer images.

The **Product Name** option is used as part of a banner at the top of the installer dialog and in various prompts for the user.

The **Default Installation Directory** is used in the installer as the default location in which to install files. The user can change the path in the installer program.

The **Installation Language** option lets you select the language that is used for messages in the resulting installer. You can select from Danish, Dutch, English, Finnish, French, German, Italian, Japanese, Norwegian, Portuguese, Spanish, and Swedish.

The **Media Size** ring option lets you specify how the file should be segmented—for 720 KB, 1.2 MB, or 1.4 MB floppies. Even if you plan to distribute the files by CD, you will have 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 **kBytes Reserved on First Disk** option 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.

The **Write Disk Images To** option lets you specify where to write the disk images on your computer. The **Create Distribution Kit** option creates a setup program in that directory as well as files named `data.001`, `data.002`, and so on. If you plan to put the disk images on

floppy, the `setup` and `data.001` files should be copied to the first floppy, the `data.002` file should be copied to the second floppy, and so on.

Files

You use the Files dialog box to specify the files that you want to install. You specify the files as a set of file groups. A file group can contain one or more files from possibly several different source locations on your machine. The application installs every file in a group to the same directory and shares the same "overwrite" setting (for example, should the files overwrite existing files of the same name without a prompt, with a prompt, only if the file being installed is newer, or never overwrite).

Press the Add Group button to define a group. You are prompted for a name for the group. This name is never seen by the user; it is only used in the **Create Distribution Kit** option to discriminate between groups. You are then prompted to select files. You can add as many files as you want. Click on the Add File button to add a single file, or click on the Add Directory Contents to add every file in a directory, not including subdirectories.

Once you have defined a group, the group shows up in the **File Groups** listbox option. When you select a group, you can specify the following settings for that group.

- The **Group Destination** option lets you specify whether you want the files in the selected group to install in the `Installation` directory or in the `Windows` directory. If you want to install the files in a subdirectory of the `Installation` directory or `Windows` directory, specify a relative path in the **Relative Path** option.
- The **Replace Existing Files** option lets you specify what should happen if a file of the same name as one of the source files is found in the destination directory. You can choose to always replace the file, ask the user, replace the file automatically if the source file is newer than the destination file, or never replace the file.

You can also edit the contents of a group using **Edit Group** or delete the group using **Delete Group**.

Program Group

You can use the Program Group dialog box to create a program group for your installation. Under Windows 3.1 and Windows NT, a program

group appears as an icon/window in the program manager. Under Windows 95, a program group shows up in **Start»Programs** as a submenu. The program group contains icons for the files you select, serving as shortcuts that make it easy to start the program regardless of its location on your computer.

Regardless of whether you plan to add items to a program group, you should specify a program group name in the **Program Group** field. Even if you don't choose to install any files in a program group, an uninstall icon is added to the group under Windows 3.1 and NT (under Windows 95, you use the **Add/Remove Programs** control panel to uninstall programs).

Click on **Add File** to add a file to the program group. A dialog box appears showing you the currently defined file groups. If you select a file group, you can then select a file within that group.

Advanced

The Advanced dialog box contain options that let you customize the behavior of the installer. These options are intended for fairly isolated applications, so you typically should not need to use them.

Select the **Use Custom Script** option if you want to change to look of the installer. The LabVIEW and LabWindows Create Distribution Kit installers are based upon licensed components from Helpful Programs Incorporated (HPI) **Create Distribution Kit** uses a file called `template.inf`, which is written in the HPI InstallIt scripting language and describes the dialogs that appear during the installation. If you need to customize the look of the installer, you could make a copy of this file, make the necessary modifications, and then specify the location of the modified script using this option.

We generally recommend against using this option, since the template script is written in a fairly flexible but somewhat complex scripting language. National Instruments has some technical information available concerning a few common modifications you might make to an installer script (many of them are LabWindows®/CVI technical notes, because both LabWindows/CVI and LabVIEW use the same licensed components for the **Create Distribution Kit** options). Call National Instruments to order these technical notes. If you plan to customize the look of the installer beyond the scope of those very simple modifications, you will need to buy the InstallIt package from

HPI in order to get the documentation and support for questions related to the scripting language.

Select the **Run Executable After Installation** option if you want to run a program after the installation completes. Additionally, you can use this option 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. You could 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.

If you choose to run an executable after the installation completes, you can use the **Command Line Arguments** to specify arguments that should be passed to the program. In addition to specifying standard arguments, you can embed any of the following options 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.

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 a preferences 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 file name, which will be the application name instead

of `labview`. Remember to include your preference file (`.ini`) with your application.This preference file should have the same root name as your application plus `.ini` (for example, `simple.exe` would store its preferences in `simple.ini`). You also must edit the new file and change the first line from `LabVIEW` to `your executable name`.

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.