LabVIEW™ Application Builder

Version 5.0

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 5.0 with the LabVIEW 5.0 Development System.

Contents

321809A-01 *January 1998*

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

(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.

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.

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 for Sun runs on SPARCstations with SunOS 4.1.3 (Solaris 1.1) or later, and Solaris 2.4 or later. LabVIEW for HP-UX runs on Hewlett-Packard Model 9000 Series 700 computers with HP-UX 9.0.5 or later. LabVIEW for Concurrent PowerMAX runs on PowerMAX version 4.1 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 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

You do not need any other patches to run LabVIEW on Solaris 2.4 or later.

Operating System Patches on Concurrent PowerMAX

You need at least the following patches to run LabVIEW on Concurrent PowerMAX 4.1. The base patches are odd numbers only.

patch	base-001	Base System Patch 001
patch	base-003	Base System Patch 003
patch	base-005	Base System Patch 005
patch	base-007	Base System Patch 007
patch	base-009	Base System Patch 009

Installing the LabVIEW Application Builder Libraries

If you are upgrading from an older version of the Application Builder, install the 5.0 libraries over your old ones.

Windows

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).

1. Run `setup.exe` on your Application Builder disk.
2. 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 disks 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 LabVIEW Application Builder for Macintosh package contains disks for installing the libraries on 680*x*0-based or Power Macintoshes.

1. Insert the Application Builder disk and double-click the LabVIEW AppLibs Installer icon.
2. 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 disks 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`

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.

Verifying Installation of Application Builder Libraries

If you launch LabVIEW after installing the Application Builder Libraries, choose **Project»Build Application...**. If this item appears grayed out, verify 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.

In addition, 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.

Windows

The **Project** menu has a **Create Distribution Kit** item. After you build an application, use this item if you want to create an installer for your application. Additionally, **Create Distribution Kit** automatically splits your application into files that can fit on floppies.

Changes to the Application Builder Libraries

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 *Customizable Features* section in this document.

Changes Introduced between Versions 4.0 and 4.1

The following list contains a bug fix and a change made between versions 4.0 and 4.1:

- If the installed libraries are read-only, building an application fails.
- **(Windows)** The **Create Distribution** item uses a newer install program. This new version fixes several floppy installation problems and uninstaller problems.

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.
- **(Windows)** Installers built with the Create Distribution Kit did not create the correct program group name.

- **(Windows 3.1)** Uninstalling a distribution built with Create Distribution Kit might have produced a script error.
- **(Windows)** Canceling the **Load Setup** and **Save Setup** options in the Create Distribution Kit did not work correctly.

Changes Introduced between Versions 3.1 and 4.0

The following new feature was added between versions 3.1 and 4.0:

(Windows) A new **Create Distribution Kit** item in the **Project** menu makes it easy for you to distribute your VIs. Selecting this item 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 Versions 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 this document.
- **(Windows)** When you build an application, you can select whether the application **File** menu has an **Open** menu item and an **Exit...** menu item. If you remove the **Exit...** item, your VIs use the Quit LabVIEW function to end the application.
- **(Macintosh and UNIX)** When you build an application, you can select whether the application **File** menu has an **Open** menu item and a **Quit** menu item. If you remove the **Quit** item, your VIs use the Quit LabVIEW function to end the application.
- LabVIEW applications can call VIs outside of themselves. When you embed VIs, it is not necessary to embed every VI. Also, you can use the VI Server functions to dynamically call VIs not embedded in the application.
- **(Macintosh)** You can use the **Project»Build Application...** command to specify the default memory partition of the application (the amount you see when you select the application in the finder and select **Show VI Info…**).
- **(Windows and UNIX)** LabVIEW uses memory in the same way as other applications. As a result, it is no longer necessary to specify the `appTotalMem` or `totalMemSize` preferences.

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 items 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.

Customizable Features

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

- 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 you can use the same run-time engine for multiple sets of VIs. This reduces the disk space usage for a customer running multiple VIs.

  Additionally, you can use a combination of these two solutions. If you embed VIs within a library, they can still call subVIs 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 require upgrading 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 the subVIs can be used in another development system, because the users can view the diagrams.

- **(Windows 95/NT)** Do you want the application to be an ActiveX server?

  If you enable the **ActiveX Server Support** option, your application responds 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 **ProgID Prefix** 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 ProgIDs. For example, if you specify the **ProgID Prefix** as `myapp`, clients instantiate an application object using the progID `myapp.application`.

- 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 an About VI, the application has a basic, default About dialog box. For more information, refer to the *Create and Save an About VI* section in this document.

How to Build an Application

This section describes how to build an application. If you want your top-level VIs to run when opened, select **Run When Opened** from the VI Setup dialog of the top-level VI. If you want all of your VIs embedded within the application, save your application VIs into a single VI library by selecting **File»Save with Options**.

If you click the **Application Distribution** item, 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 and run-time menus referenced by the VIs in the VI library.

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

Besides saving all the VIs necessary for your application in a VI library, you can also save a VI 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). More information on the About VI is in the next section of this document, *Create and Save an About VI*.

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 item and an About VI is installed, that VI runs. 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.***

Save VIs that Use VI Server Functions

VI Server functions dynamically load into applications; therefore, they are not part of the hierarchy of the top-level VIs and require a different procedure for saving. If you build an application that contains VI Server functions, use one of the following methods for saving:

- The easiest method is to save all dynamically loaded VIs into a single library:
  - Build a dummy VI whose diagram contains the top-level application VI and all the dynamic top-level VIs. When you save this VI with the options for Application Distribution, it saves all the necessary VIs as well. Delete the dummy VI from the application library at this point, if you choose. The paths to the dynamic VIs might change and it might be necessary to modify your code to account for this change.
  - Select **File»Save With Options...»Application Distribution...** and save all dynamically loaded VIs to a new library, as you normally do. Then, for each VI that is dynamically loaded, repeat this process, but save to the same library as your main application. This includes the Hierarchies of your dynamically called applications.

  From a file management standpoint, this method can create a large executable file. From a memory standpoint, you still benefit from dynamic loading because the entire executable file does not load into memory.

- The second method is to embed only your main application within the executable and dynamically call external applications as separate files (VIs and/or LLBs). If you use VI Server functions, you then must

specify the path to the dynamically called applications by using the function Current VIs Path, followed by a Strip Path.

Use the Build Application Item

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

If you want to embed a VI library, click the **embed** menu item. 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, on the Windows 95/NT platform, you can choose whether the application acts as an ActiveX server.

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** item, LabVIEW displays the **Edit VI Library...** dialog box so you can select which VIs open at launch time. Use the **Top Level** item to mark the VIs 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.

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. Choose **File»Save with Options...»Application Distribution»Save**. When prompted, enter the name `sample.llb` and then click **Select**.
4. 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.

5. In the **edit** mode, select **VI Setup...** in the pop-up menu of the icon pane of the Sample VI. Then select the **Run When Opened**»**OK**.
6. Save a copy of the About Sample VI into `examples\sample.llb` and click **OK**.
7. Now you can build the application. Select **File**»**Build Application…**. Click the **Embed VI Library** button and choose `sample.llb` and click **OK**. LabVIEW then prompts you to mark which VIs you want opened when the application launches. 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 **(Windows)**`sample.exe` or **(Macintosh** and **UNIX)** `sample`.

☞ **Note** **(Windows and UNIX)** ***If you do not build the application at the top-level of the LabVIEW directory, you must place several files in the same directory as the application. These files communicate with hardware. See the*** *Additional Files Required by Applications* ***section of this document for a list of the files to be stored with your application.***

8. Quit LabVIEW, and run the **(Windows)** `sample.exe` or **(Macintosh and UNIX)** `sample` application. It launches and then automatically opens and runs the Sample VI. Look at the menu items that are now available. Select **(Windows**, **Macintosh**, **and UNIX) Help»About…** or **(Macintosh) Apple»About…**. When you finish, click the QUIT button on the front panel of the application.
9. In practice, you might 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 might not be the desired behavior. It is best to structure most applications so that the **Abort** item is disabled on the top-level VI, and the VI has a **Quit** item 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 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 serial port or data acquisition functionality include the `serpdrv` or `daqdrv` files. If your application uses a GPIB or data

acquisition board the user must install the hardware drivers that come with their boards.

(Windows 95/NT) If you created an ActiveX type library for your application, the type library and help file must be included.

(Windows 95/NT) 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.

(Windows 95/NT) 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.

(Windows 3.1) You must include `lvdev5.dll`. This file can be found in the `resource` directory of your LabVIEW installation.

(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.

Distribution Rights

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

Packaging Your Files for Distribution

A LabVIEW application can be 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 account for at least half of the size of your VIs. Even so, most run-time applications do not fit on a single floppy disk. To distribute your applications, you can put them on a larger capacity medium, such as a CD or magnetic tape, or you can compress the applications and create an installer for them.

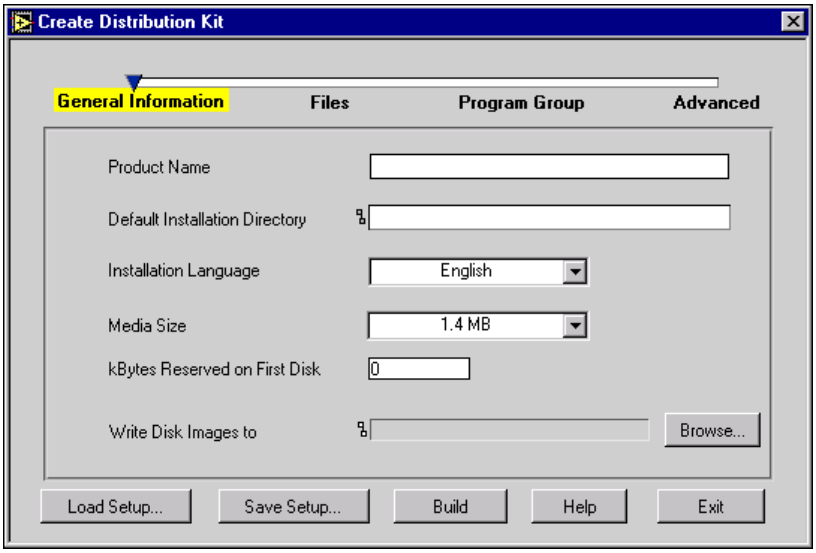
(Windows) The Windows version of the Application Builder Libraries adds a **Create Distribution Kit** item to the **Project** menu. This command makes it simple to create professional installers for your LabVIEW applications.

(Macintosh) There are two shareware compression utilities, Compact Pro and Stuffit, widely available from online services. You can use these compression utilities to create self-extracting archives, which decompress themselves when you double-click them.

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

Create Distribution Kit

(Windows) When you select Project»Create Distribution Kit the following dialog box appears.



To create an installer, enter the information in this dialog box, then click the **Build** button at the bottom. This command compresses the files, builds an installer, and writes the disk images to the hard drive path specified in **Write Disk Images to**. 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 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** item to save the information that you enter in this dialog box. Subsequently, you can use the **Load Setup** item to retrieve information.

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

General Information

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

The **Product Name** item 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** item is used in the installer as the default location for installed files. The user can change the path in the installer program.

The **Installation Language** item 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 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 **kBytes Reserved on First Disk** 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.

The **Write Disk Images to** item lets you specify where to write the disk images on your computer. The **Create Distribution Kit** item 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, 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.

Files

You use the Files dialog box to specify the files that you want to install. Specify the files as a set of file groups. A file group can contain one or more files from several possible source locations on your computer. The application installs every file in a group to the same directory and shares the same overwrite setting. For example, you can choose whether 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** item to discriminate between groups. You are then prompted to select files. You can add as many files as you want. Click the **Add File** button to add a single file, or click **Add Directory Contents** to add every file in a directory, not including subdirectories.

Once you define a group, it shows up in the **File Groups** listbox item. When you select a group, you can specify the following settings for that group.

- The **Group Destination** item 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** item.
- The **Replace Existing Files** item lets you specify what happens 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 contents 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, it is best to 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 **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 contains items that let you customize the behavior of the installer. These items are intended for fairly isolated applications, so typically it is not necessary to use them.

Select the **Use Custom Script** item 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. To customize the look of the installer, make a copy of this file, make the necessary modifications, and then specify the location of the modified script using this item.

However, because the template script is written in a fairly flexible but somewhat complex scripting language, using this item is not recommended. 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** items). 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, it is necessary to buy the InstallIt package from HPI in order to get documentation and support for questions related to the scripting language.

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.

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.

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.

(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.

(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>`. 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.

You might want to disable the following three items: **Abort**, **Close Box**, and **Free Run**. You also might 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, it is best to 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 a common error that occurs when you launch a LabVIEW application.

Table 1. Application Launch Error

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.

321809A-01 Jan98